

The data arrangement problem on binary trees

ERANDA ÇELA*

JOACHIM SCHAUER**

ROSTISLAV STANĚK**

Abstract

The *data arrangement problem on regular trees (DAPT)* consists in assigning the vertices of a given graph G , called the guest graph, to the leaves of a d -regular tree T , called the host graph, such that the sum of the pairwise distances of all pairs of leaves in T which correspond to the edges of G is minimised. LUCZAK and NOBLE [8] have shown that this problem is \mathcal{NP} -hard for every fixed $d \geq 2$. In this paper we show that the DAPT remains \mathcal{NP} -hard even if the guest graph is a tree, an issue which was posed as an open question in [8].

We deal with a special case of the DAPT where both the guest and the host graph are binary regular trees and provide a 1.015-approximation algorithm for special case. The solution produced by the algorithm and the corresponding value of the objective function are given in closed form. The analysis of the approximation algorithm involves an auxiliary problem which is interesting on its own, namely the *k-balanced partitioning problem (k-BPP)* for binary regular trees and particular choices of k . We derive a lower bound for the later problem and obtain a lower bound for the original problem by solving h_G instances of the k -BPP, where h_G is the height of the host graph G .

Keywords. Graph embedding; data arrangement problem; approximation algorithm; partitioning; k -balanced partitioning problem; binary trees

1 Introduction

Given an undirected graph $G = (V(G), E(G))$ with $|V(G)| = n$, an undirected graph $H = (V(H), E(H))$ with $|V(H)| \geq n$ and some subset B of the vertex set of H , $B \subseteq V(H)$ with $|B| \geq n$, the **generic graph embedding problem (GEP)** consists of finding an injective embedding of the vertices of G into the vertices in B such that some prespecified objective function is minimised. Throughout this paper we will call G **the**

*cela@opt.math.tu-graz.ac.at. Department of Optimization and Discrete Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria

**{joachim.schauer, rostislav.stanek}@uni-graz.at. Department of Statistics and Operations Research, University of Graz, Universitaetsstrasse 15, A-8010 Graz, Austria

guest graph and H **the host graph**. A commonly used objective function maps an embedding $\phi: V(G) \rightarrow B$ to

$$\sum_{(i,j) \in E(G)} d(\phi(i), \phi(j)), \quad (1)$$

where $d(x, y)$ denotes the length of the shortest path between x and y in H . The host graph H may be a weighted or a non-weighted graph; in the second case the path lengths coincide with the respective number of edges. Given a non-negative number $A \in \mathbb{R}$, the decision version of the GEP asks whether there is an injective embedding $\phi: V(G) \rightarrow B$ such that the objective function does not exceed A .

Different versions of the GEP have been studied in the literature; the **linear arrangement problem**, where the guest graph is a one dimensional equidistant grid with n vertices is probably the most prominent among them (see CHUNG [3], JUVAN and MOHAR [6], SHILOACH [9]).

This paper deals with the version of the GEP where the guest graph G has n vertices, the host graph H is a complete d -regular tree of height $\lceil \log_d n \rceil$ and the set B consists of the leaves of H . From now on we will denote the host graph by T . The height of T as specified above guarantees that the number $|B|$ of leaves fulfills $|B| \geq n$ and that the number of the direct predecessors of the leaves in T is smaller than n . Thus $\lceil \log_d n \rceil$ is the smallest height of a d -regular tree which is able to accommodate an injective embedding of the vertices of the guest graph on its leaves. This problem is originally motivated by real problems in communication systems and was first posed by LUCZAK and NOBLE [8].

We call the above described version of the GEP the **data arrangement problem on regular trees (DAPT)**. LUCZAK and NOBLE [8] have shown that the DAPT is \mathcal{NP} -hard for every fixed $d \geq 2$ and have posed as an open question the computational complexity of the DAPT in the case where the guest graph is a tree. In this paper we answer this question and show that this particular case of the problem is \mathcal{NP} -hard for every $d \geq 2$. In the special case where both the guest graph G and the host graph T are binary regular trees we give a 1.015-approximation algorithm.

The paper is organised as follows. Section 2 discusses some general properties of the problem and introduces the notation used throughout the paper. Section 3 presents an algorithm for the DAPT on regular binary trees, where the guest graph is also a regular binary tree. Section 4 deals with the *k-balanced partitioning problem (k-BPP)* in binary regular trees. This version of the *k-BPP* serves as an auxiliary problem in the sense that it leads to a lower bound for the objective function value of the DAPT on regular binary trees. In Section 5 we use the auxiliary problem and the lower bound mentioned above to analyze the algorithm presented in Section 3 and show that the latter is an 1.015-approximation algorithm. In Section 6 it is proven that the DAPT is \mathcal{NP} -hard for every $d \geq 2$ even if the guest graph is a tree. Finally, in Section 7 we provide some final notes, conclusions and questions for further research.

2 Notations and general properties of the DAPT

First, we formally define a **d -regular tree** as follows:

Definition 1 (d -regular tree). A tree $T = (V(T), E(T))$ is called a **d -regular tree**, $d \in \mathbb{N}$, $d \geq 2$, if

1. it contains a specific vertex $v_1 \in V$ of degree d which is called the **root** of T and is also denoted by $r(T)$,
2. every vertex but the leaves and the root has degree $d + 1$ and
3. there is a number $h \in \mathbb{N}$ such that the length $d(l, v_1)$ of the path between the root v_1 and a leaf l equals h for every leaf l of T .

The number h is called the **height** of the tree T , and is also denoted by $h(T)$. For every vertex $v \in V \setminus \{v_1\}$, i.e. for any vertex v but the root v_1 , the unique neighbour of v in the path between v_1 and v in T is called the **father** of v . All other neighbours of v (if any) are called the **children** of v . The neighbours of the root v_1 are called **children** of v_1 . The **level of a vertex** v , denoted by $\text{level}(v)$, is the length (i.e. the number of edges) of the unique path joining v and the root v_1 of the tree. Thus in a d -regular tree of height h the level of each leaf equals h , whereas the level of the root v_1 equals 0. All vertices w , $w \neq v$, of the unique path joining v and the root v_1 of the tree are called **ancestors** of v . Given two vertices v and u their **most recent common ancestor** w is their common ancestor with the highest level, i.e. $w = \text{argmax}\{\text{level}(t) : t \text{ is a common ancestor of } v \text{ and } u\}$.

A **subtree of k -th order** of a d -regular tree T is a d -regular subtree T' of T of height $h(T') = h(T) - k$, rooted at some vertex of level k in T . A subtree of first order will be called a **basic subtree**.

Consider a guest graph $G = (V, E)$ with n vertices, and a host graph T which is a d -regular tree of height h , $h := \lceil \log_d n \rceil$. Let B be the set of leaves of T . Notice that due to the above choice of h we get the following upper bound for the number $b = |B|$ of leaves:

$$b := |B| = d^h = d^{h-1}d < nd. \quad (2)$$

Definition 2 (data arrangement problem on regular trees). Given a guest graph $G = (V, E)$ with $|V| = n$ and a host graph T which is a d -regular tree with set of leaves B and height equal to $\lceil \log_d n \rceil$, an **arrangement** is an injective mapping $\phi: V \rightarrow B$. The **data arrangement problem on regular trees (DAPT)** asks for an arrangement ϕ that minimises the objective value $OV(G, d, \phi)$

$$OV(G, d, \phi) := \sum_{(u,v) \in E} d_T(\phi(u), \phi(v)), \quad (3)$$

where $d_T(\phi(u), \phi(v))$ denotes the length of the unique $\phi(u)$ - $\phi(v)$ -path in the d -regular tree T . Such an arrangement is called an **optimal arrangement**. The corresponding

value of the objective functions is called the **optimal value** of the problem. An instance of the DAPT is fully determined by the guest graph and the parameter d of the d -regular tree T which serves as a host graph. Such an instance of the problem will be denoted by $DAPT(G, d)$ and its optimal value will be denoted by $OPT(G, d)$.

Theorem 3. *The DAPT is \mathcal{NP} -hard for every fixed $d \geq 2$.*

Proof. See LUCZAK and NOBLE [8]. □

Example 4. A guest graph G of height 3 is shown in Figure 1. Figure 2 represents the same guest graph G , but with another colouring of its vertices; the role of the colouring will be explained below. Figures 3 and 4 depict a feasible ϕ arrangement and an optimal arrangement ϕ_A of G , yielding the objective function values $OV(G, 2, \phi) = 58$ and $OV(G, 2, \phi_A) = 56$, respectively.

Note that the labels in the vertices of the guest graphs denote the index of the vertices in the so-called canonical ordering (defined below). The labels of the leaves in the host graphs represent the arrangement: the label of each leaf coincides with the index of the vertex arranged at that leaf (in the canonical ordering).

The colours should help to capture some properties of the arrangement at a glance: the set of vertices of a certain colour in the guest graph is arranged at the set of leaves of the same colour in the host graph T . Moreover, some of the vertices in the guest graph have a dashed boundary, the others have a solid boundary. The graphical representation of an arrangement preserves the boundary property, in the sense that vertices with a dashed boundary in G are arranged at dashed-boundary leaves of the same colour in T .

Definition 5 (canonical order). *The canonical order of the vertices of the guest graph and the canonical order of the leaves of the host graph are defined recursively as follows.*

- (a) *The **canonical order of the leaves of a d -regular tree T** is an arbitrary but fixed order if $h(T) = 1$. If $h(T) > 1$ then an order of the leaves is called canonical if (i) it implies a canonical order of the leaves of every basic subtree of T , and (ii) for an arbitrary but fixed order of the children ch_1, \dots, ch_d of the root $r(T)$ of T all leaves of the basic subtree rooted at ch_i precede all leaves of the basic subtree rooted at ch_j , for $i < j$, $i, j \in \{1, 2, \dots, d\}$, in this order.*
- (b) *A **canonical ordering of the vertices of a d -regular tree T** is the unique order if $h(T) = 0$. If $h(T) \geq 1$, a canonical order of the vertices of T is an order obtained by extending the canonical order of the vertices of the d -regular tree T' of height $h(T') = h(T) - 1$ obtained from T by removing all of its leaves and fulfilling the following two properties: (i) all vertices of T' precede the leaves of T , and (ii) for any two leaves a and b of T' , if a precedes b , then all children of a in T precede all children of b in T .*

If the leaves of a d -regular tree T are ordered according to the canonical order as above, then the pairwise distances between them are given by a simple formula.

Observation 6. Let T be a d -regular tree of height $h := h(T)$ and let its b leaves be labelled according to the canonical order $b_1 \prec b_2 \prec \dots \prec b_b$. Then the distances between the leaves in T are given as $d_T(b_i, b_j) = 2l$, where

$$l := \min \left\{ k \in \{1, 2, \dots, h\} : \left\lfloor \frac{i-1}{d^k} \right\rfloor = \left\lfloor \frac{j-1}{d^k} \right\rfloor \right\}, \quad (4)$$

for all $i, j \in \{1, 2, \dots, b\}$. If vertex u is the most recent common ancestor of b_i and b_j , then $h - l = \text{level}(u)$.

Proof. See ÇELA and STANĚK [2]. □

In this paper we deal with the special case where both the guest graph G and the host graph T are binary regular trees; an instance of this problem is fully specified by the guest graph G and will be denoted by $DAPT(G, 2)$. From now on we denote by h_G the height of the guest graph G and by h the height of the host graph T . Moreover we will always use the canonical order $v_1 \prec v_2 \prec \dots \prec v_n$ of the vertices v_i , $1 \leq i \leq n$, $n := |V(G)|$, of the guest graph, and the canonical order $b_1 \prec b_2 \prec \dots \prec b_b$ of the b leaves of the host graph T as in the observation above. See e.g. Figure 1 for an illustration of the canonical order of the vertices of a regular tree of height 3; for simplicity we specify the indices i , $1 \leq i \leq 15$ instead of the labels v_i , $1 \leq i \leq 15$.

In the following we list some obvious equalities which will be used through the rest of this paper.

Observation 7.

$$n = 2^{h_G+1} - 1 \quad (5)$$

$$h = \lceil \log_2 (2^{h_G+1} - 1) \rceil = h_G + 1 \quad (6)$$

$$b = 2^{h_G+1} = 2 \cdot 2^{h_G} = n + 1 \quad (7)$$

3 An approximation algorithm

In this section we describe an approximation algorithm \mathcal{A} for the $DAPT(G, 2)$ where the guest graph G is a binary regular tree. Later in Section 5 it will be shown that this is an α -approximation algorithm with $\alpha = \frac{203}{200}$, i.e. $OV(G, 2, \phi_A) \leq \alpha OV(G, 2, \phi_*)$ holds for every binary regular tree G , where ϕ_* denotes the optimal arrangement of $DAPT(G, 2)$ and ϕ_A denotes the arrangement computed by algorithm \mathcal{A} described below.

An approximation algorithm

Require: binary regular tree $G = (V, E)$ of height h_G whose vertices are labelled according to the canonical order

Ensure: arrangement ϕ_A

1: $b := 2^{h_G+1}$;

```

2: if  $h_G = 0$  then
3:    $\phi_A(v_1) := b_1$ ;
4: else [ $h_G > 0$ ]
5:   solve the problem for the basic subtrees  $\widehat{G}_1$  and  $\widehat{G}_2$  of height  $\widehat{h}_G = h_G - 1$  and
   obtain the respective arrangements  $\widehat{\phi}_A^{(1)}$  and  $\widehat{\phi}_A^{(2)}$ ;
6:   arrange the vertices of the left basic subtree on the leaves  $b_1, b_2, \dots, b_{\frac{1}{2}b}$  according
   to the arrangement  $\widehat{\phi}_A^{(1)}$  and the vertices of the right basic subtree on the leaves
    $b_{\frac{1}{2}b+1}, b_{\frac{1}{2}b+2}, \dots, b_b$  according to the arrangement  $\widehat{\phi}_A^{(2)}$ ;
7:    $\phi_A(v_1) := b_{\frac{1}{2}b}$ ;
8:   if  $h_G$  is odd and  $h_G \geq 3$  then
9:     exchange the vertices arranged on the leaves  $b_{\frac{1}{4}b-1}$  and  $b_{\frac{1}{2}b}$  (pair-exchange);
10:  end if
11: end if
12: return  $\phi_A$ ;

```

Algorithm 1: The approximation algorithm \mathcal{A} computes the arrangement ϕ_A .

In the following we apply this algorithm on an instance of $DAPT(G, 2)$ with $h_G = 3$. Observe that the leaf $b_{\frac{1}{2}b}$ is always free prior to the execution of pseudocode line 7 due to the recursion and due to the assignment in pseudocode line 3.

Example 8. Consider the guest graph $G = (V, E)$ of height $h_G = 3$ depicted in Figure 1 and apply algorithm \mathcal{A} .

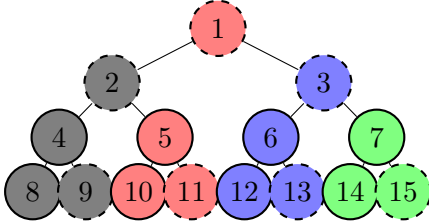


Figure 1: A guest graph $G = (V, E)$ (binary regular tree of height $h_G = 3$). The colours are related to the arrangement ϕ depicted in Figure 3.

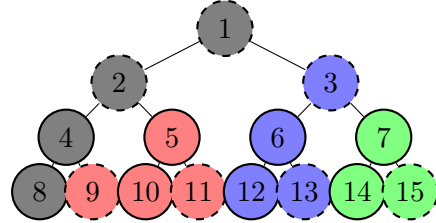


Figure 2: A guest graph $G = (V, E)$ (binary regular tree of height $h_G = 3$). The colours are related to the arrangement ϕ depicted in Figure 4.

Since $h_G = 3 > 0$, the algorithm executes the else part beginning in pseudocode line 4. In pseudocode lines 5 and 6 the arrangements for both basic subtrees, i.e. for graphs of height $\widehat{h}_G = h_G - 1 = 3 - 1 = 2$ are computed. (The arrangement $\widehat{\phi}_A$ for $h_G = 2$ is depicted in Figure 18 in Appendix.) In the next step, the root is arranged at the middle leaf (see pseudocode line 7) and the arrangement ϕ depicted in Figure 3 is obtained. The

label of each leaf corresponds to the index of the vertex of the guest graph arranged at that leaf. The objective value which corresponds to arrangement ϕ is $OV(G, 2, \phi) = 58$.

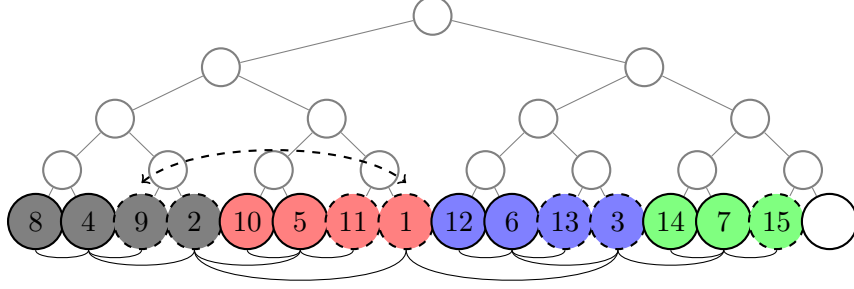


Figure 3: Arrangement ϕ with objective function value $OV(G, 2, \phi) = 58$.

Next consider the condition in pseudocode line 8: Since $h_G = 3$ is odd and $h_G = 3 \geq 3$, the pair-exchange marked in Figure 3 by arrows is performed. The value of the objective function corresponding to the resulting arrangement ϕ_A in Figure 4 is $OV(G, 2, \phi_A) = 56$. The guest graph coloured according to this arrangement is depicted in Figure 2. In fact, this arrangement is optimal, but in general Algorithm 1 does not yield an optimal arrangement.

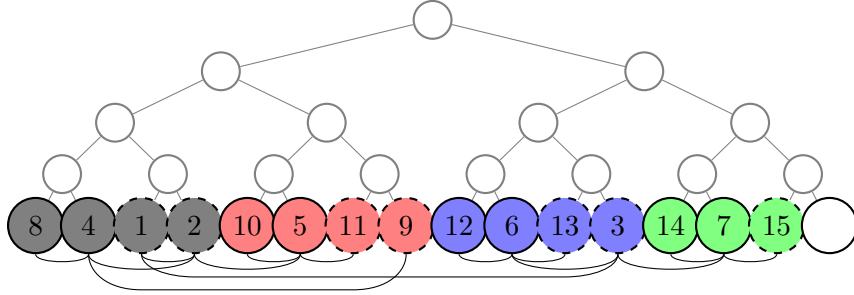


Figure 4: Arrangement ϕ_A obtained from Algorithm 1 with objective function value $OV(G, 2, \phi_A) = 56$.

Next we give a closed formula for the objective function value corresponding to the arrangement ϕ_A computed by the algorithm \mathcal{A} .

Lemma 9. *Let the guest graph $G = (V, E)$ and the host graph T be binary regular trees of heights h_G and $h = h_G + 1$ respectively, where h_G is odd, $h_G \geq 3$. Then the pair-exchange defined in Algorithm 1 in pseudocode line 9 decreases by 1 the number of edges which contribute to the objective by 4, increases by 1 the number of edges which contribute to the objective value by 2, and does not change the number of edges which contribute to the objective value by $2i$ for $i \geq 3$. Summarizing such a pair-exchange improves the value of*

the objective function by 2 as compared to the value corresponding to the arrangement available prior to this pair-exchange.

Proof. Let ϕ be the arrangement available prior to the pair-exchange steps done in pseudocode line 9 in Algorithm 1. Denote the arrangement obtained after the pair-exchange by ϕ_A . Consider the vertices and edges which are affected by the pair-exchange in pseudocode line 9. According to the algorithm (see pseudocode line 7) the root v_1 of G is arranged on the leaf $b_{\frac{1}{2}b}$ and its left and right children v_2 and v_3 are arranged on the leaves $b_{\frac{1}{4}b}$ and $b_{\frac{3}{4}b}$, respectively. (Recall that the pair-exchange is performed only if h_G is odd). Moreover the algorithm places the rightmost leaf, say x , of the basic subtree of G rooted at the child v_2 on leaf $b_{\frac{1}{4}b-1}$ of T . So the pair-exchange involves the vertices v_1 and x of G and $\phi_A(v_1) = \phi(x)$, $\phi_A(x) = \phi(v_1)$, $\phi_A(y) = \phi(y)$, for $y \in V \setminus \{v_1, x\}$, hold. The change Δ in the value of the objective functions corresponding to ϕ and ϕ_A , respectively, is then given as follows

$$\begin{aligned}
\Delta &:= OV(G, 2, \phi) - OV(G, 2, \phi_A) = \sum_{\substack{v \in V \setminus \{x\} \\ \{v, v_1\} \in E}} d_T(\phi(v), \phi(v_1)) + \\
&\sum_{\substack{v \in V \setminus \{v_1\} \\ \{v, x\} \in E}} d_T(\phi(v), \phi(x)) - \sum_{\substack{v \in V \setminus \{x\} \\ \{v, v_1\} \in E}} d_T(\phi_A(v), \phi_A(v_1)) - \sum_{\substack{v \in V \setminus \{v_1\} \\ \{v, x\} \in E}} d_T(\phi_A(v), \phi_A(x)) = \\
&\sum_{\substack{v \in V \setminus \{x\} \\ \{v, v_1\} \in E}} \left[d_T(\phi(v), \phi(v_1)) - d_T(\phi_A(v), \phi_A(v_1)) \right] + \\
&\sum_{\substack{v \in V \setminus \{v_1\} \\ \{v, x\} \in E}} \left[d_T(\phi(v), \phi(x)) - d_T(\phi_A(v), \phi_A(x)) \right] = \\
&\sum_{\substack{v \in V \setminus \{x\} \\ \{v, v_1\} \in E}} \left[d_T(\phi(v), \phi(v_1)) - d_T(\phi(v), \phi(x)) \right] + \\
&\sum_{\substack{v \in V \setminus \{v_1\} \\ \{v, x\} \in E}} \left[d_T(\phi(v), \phi(x)) - d_T(\phi(v), \phi(v_1)) \right]
\end{aligned}$$

Considering that v_1 has only two neighbours, namely v_2 and v_3 , and denoting by y the unique neighbour (i.e. the father) of leaf x in G we get

$$\begin{aligned}
\Delta &= d_T(\phi(v_2)\phi(v_1)) - d_T(\phi(v_2)\phi(x)) + d_T(\phi(v_3)\phi(v_1)) - d_T(\phi(v_3)\phi(x)) + \\
&d_T(\phi(y)\phi(x)) - d_T(\phi(y)\phi(v_1)) = 2h_G - 2 + 2(h_G + 1) - 2(h_G + 1) + 4 - 2h_G = 2.
\end{aligned}$$

□

Lemma 10. *Let the guest graph $G = (V, E)$ and the host graph T be binary regular trees of heights h_G and $h = h_G + 1$ respectively. Then the value $OV(G, 2, \phi_A)$ of the objective function of the DAPT corresponding to the arrangement ϕ_A obtained from Algorithm 1 is given as follows:*

$$OV(G_{h_G}, 2, \phi_A) := OV(G, 2, \phi_A) = \begin{cases} 0 & \text{for } h_G = 0 \\ \frac{29}{3} \cdot 2^{h_G} - 4h_G - 9 + \frac{1}{3}(-1)^{h_G} & \text{for } h_G \geq 1 \end{cases} \quad (8)$$

Proof. The proof is done by induction with respect to h_G . Let us denote G_h a binary regular tree of height h throughout this proof. Clearly we have $OV(G_0, 2, \phi_A) = 0$. For $h_G = 1$ we obviously have $OV(G_1, 2, \phi_A) = 2 + 4 = 6$ by the construction (see Figures 15 and 16 in Appendix). Both these equalities are consistent with (8).

Assume that (8) holds for some $h_G \geq 1$. For $h_G + 1$ we get

$$OV(G_{h_G+1}, 2, \phi_A) = \begin{cases} 2OV(G_{h_G}, 2, \phi_A) + 2(h_G + 1) + 2(h_G + 2) - 2 & \text{for } h_G + 1 \text{ odd} \\ 2OV(G_{h_G}, 2, \phi_A) + 2(h_G + 1) + 2(h_G + 2) & \text{for } h_G + 1 \text{ even} \end{cases}, \quad (9)$$

where:

- $2OV(G_{h_G+1}, 2, \phi_A)$ represents the objective function value corresponding to the arrangements of the basic subtrees.
- $2(h_G + 1)$ and $2(h_G + 2)$ represent the contribution of the edges connecting the root v_1 with its left and right child in the objective function value, respectively. (Prior to the pair-exchange step the root v_1 is arranged at the leaf $b_{\frac{1}{2}b}$ while its children, v_2 and v_3 are arranged at the leaves $v_{\frac{1}{4}b}$ and $b_{\frac{3}{4}b}$, respectively.)
- -2 represents the contribution of the pair-exchange step if $h_G + 1$ is odd ($h_G + 1 \geq 3$ since $h_G \geq 1$), according to Lemma 9.

According to the induction assumption we substitute $OV(G_{h_G}, 2, \phi_A)$ by the expression on the right hand side of equation (8) and after simplifying we get

$$OV(G_{h_G+1}, 2, \phi_A) = \begin{cases} \frac{29}{3} \cdot 2^{h_G+1} - 4(h_G + 1) - 9 + \frac{1}{3}(-1)^{h_G+1} & \text{if } h_G + 1 \text{ is odd} \\ \frac{29}{3} \cdot 2^{h_G+1} - 4(h_G + 1) - 9 + \frac{1}{3}(-1)^{h_G+1} & \text{if } h_G + 1 \text{ is even.} \end{cases}$$

□

Finally, notice that this approximation algorithm \mathcal{A} does not solve the problem to optimality as illustrated by the following example.

Example 11. Consider a guest graph $G = (V, E)$ of height $h_G = 6$ depicted in Figure 5. The arrangement ϕ_A computed by Algorithm 1 is depicted in Figures 6 and 7; it yields an objective function value of $OV(G, 2, \phi_A) = 586$. Consider now another arrangement ϕ for the same graph yielding an objective function value of $OV(G, 2, \phi) = 584$ and depicted explicitly in Figures 8, 9 and 10. Later in Section 5 we will show that the approximation algorithm \mathcal{A} yields an optimal arrangement ϕ_A for $h_G \leq 5$. Thus this is the smallest instance of the $DAPT(G, 2)$ for which the algorithm \mathcal{A} does not compute an optimal arrangement.

4 The k -balanced partitioning problem

In this section we introduce the k -balanced partitioning problem and a special case of it which will be involved in the analysis of the approximation algorithm for the $DAPT(G, 2)$ with a binary regular tree G .

Definition 12 (k -balanced partitioning problem). Given a graph $G = (V, E)$ with $|V| = n$ and $k \geq 2$, a **k -balanced partition** is a partition of the vertex set V into k non-empty partition sets $V_1 \neq \emptyset, V_2 \neq \emptyset, \dots, V_k \neq \emptyset$, where $\cup_{i=1}^k V_i = V$, $V_i \cap V_j = \emptyset$ for every $i \neq j$ and $|V_i| \leq \lceil \frac{n}{k} \rceil$ for all $1 \leq i \leq k$. The **k -balanced partitioning problem (k -BPP)** asks for a k -balanced partition \mathcal{V} which minimises

$$c(G, \mathcal{V}) := \left| \{ (u, v) \in E \mid u \in V_i, v \in V_j, i \neq j \} \right|, \quad (10)$$

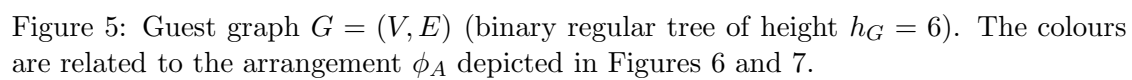
where $\mathcal{V} := \{V_i \mid 1 \leq i \leq k\}$.

k -BPP is a well known \mathcal{NP} -hard problem (for $k = 2$ we get the *minimum bisection problem* which is \mathcal{NP} -hard, see GAREY and JOHNSON [5]). A lot of work has been done focusing on the computational complexity of the k -BPP. ANDREEV and RÄCKE proved further complexity results for a generalization allowing near-balanced partitions [1]. KRAUTHGAMER, NAOR and SCHWARTZ provide an approximation algorithm achieving an approximation of $O(\sqrt{\log n \log k})$ [7]. And finally, FELDMANN and FOSCHINI proved that the k -BPP remains APX -hard even if the graph G is restricted to be an unweighted tree with constant maximum degree [4].

We deal with a special case of this problem where $G = (V, E)$ is a binary regular tree of height $h \geq 1$ and where $k = 2^{k'}$ and $1 \leq k' \leq h$. The following facts are obvious:

Observation 13. Let $G = (V, E)$ be a binary regular tree of height $h \geq 1$ with $n = 2^{h+1} - 1$ vertices. Let $\mathcal{V} = \{V_1, V_2, \dots, V_k\}$ be a k -balanced partition with $k = 2^{k'}$ and $1 \leq k' \leq h$. Then one of the partition sets in \mathcal{V} has $n_s := \frac{n+1}{k} - 1$ elements and is called the **small partition set**. All other partition sets have $n_b := \frac{n+1}{k}$ elements and are called **big partition sets**. Moreover the following equalities clearly hold

$$n_s = 2^{h-k'+1} - 1 \text{ and } n_b = 2^{h-k'+1}. \quad (11)$$



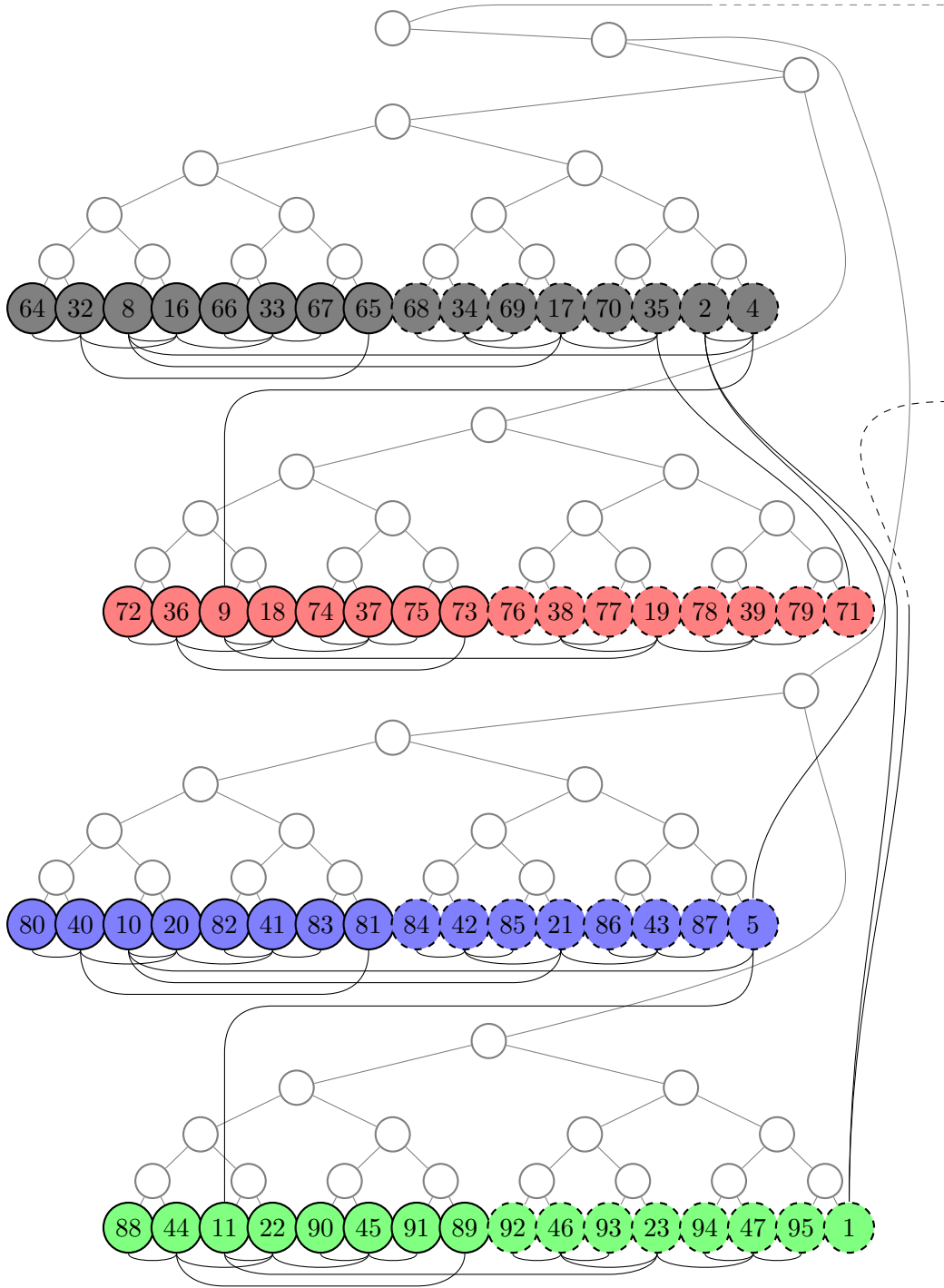


Figure 6: Arrangement ϕ_A obtained from Algorithm 1 for the guest graph $G = (V, E)$ depicted in Figure 5 – first part. Its objective function value is $OV(G, 2, \phi_A) = 586$.

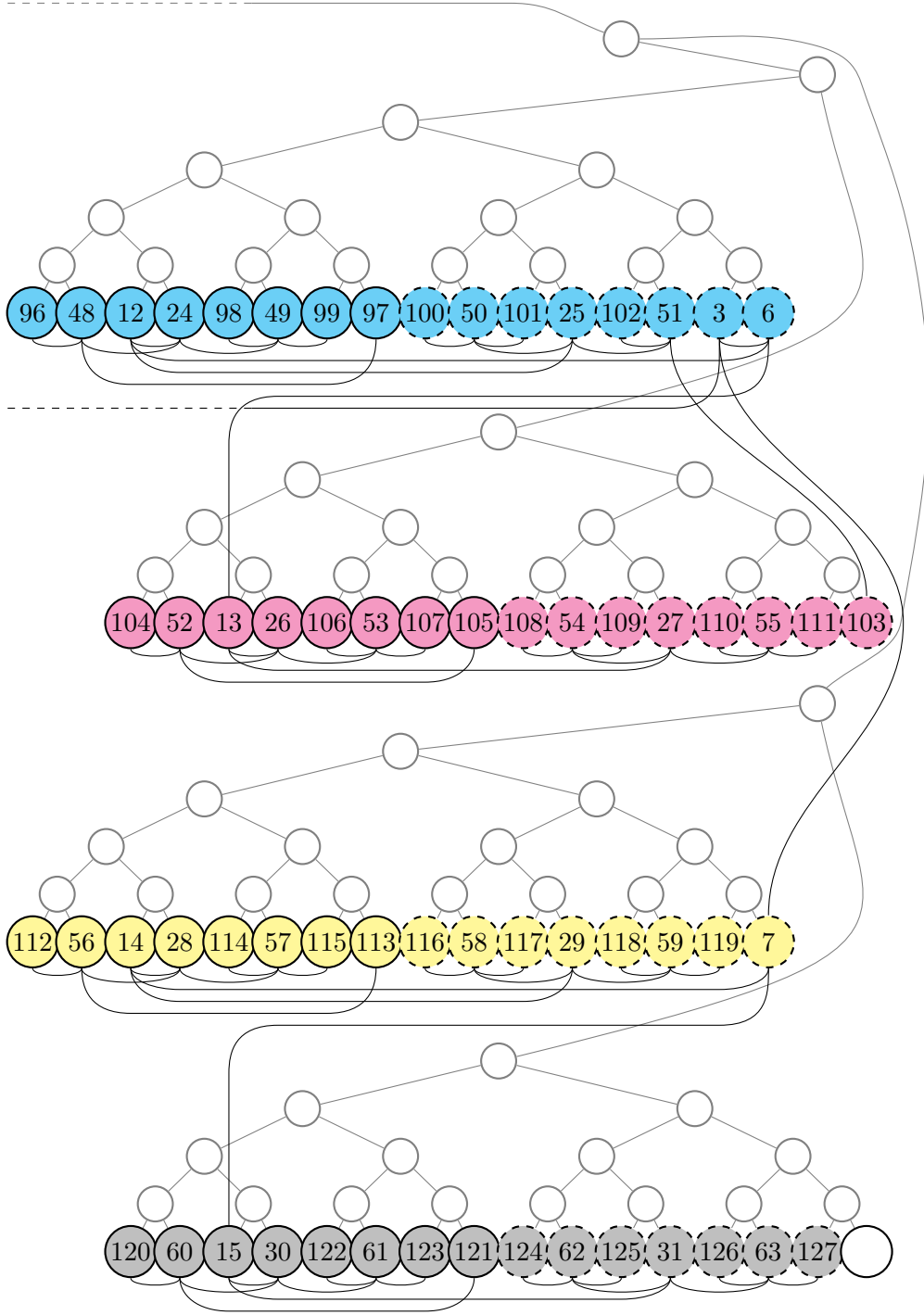


Figure 7: Arrangement ϕ_A obtained from Algorithm 1 for the guest graph $G = (V, E)$ depicted in Figure 5 – second part. Its objective function value is $OV(G, 2, \phi_A) = 586$.

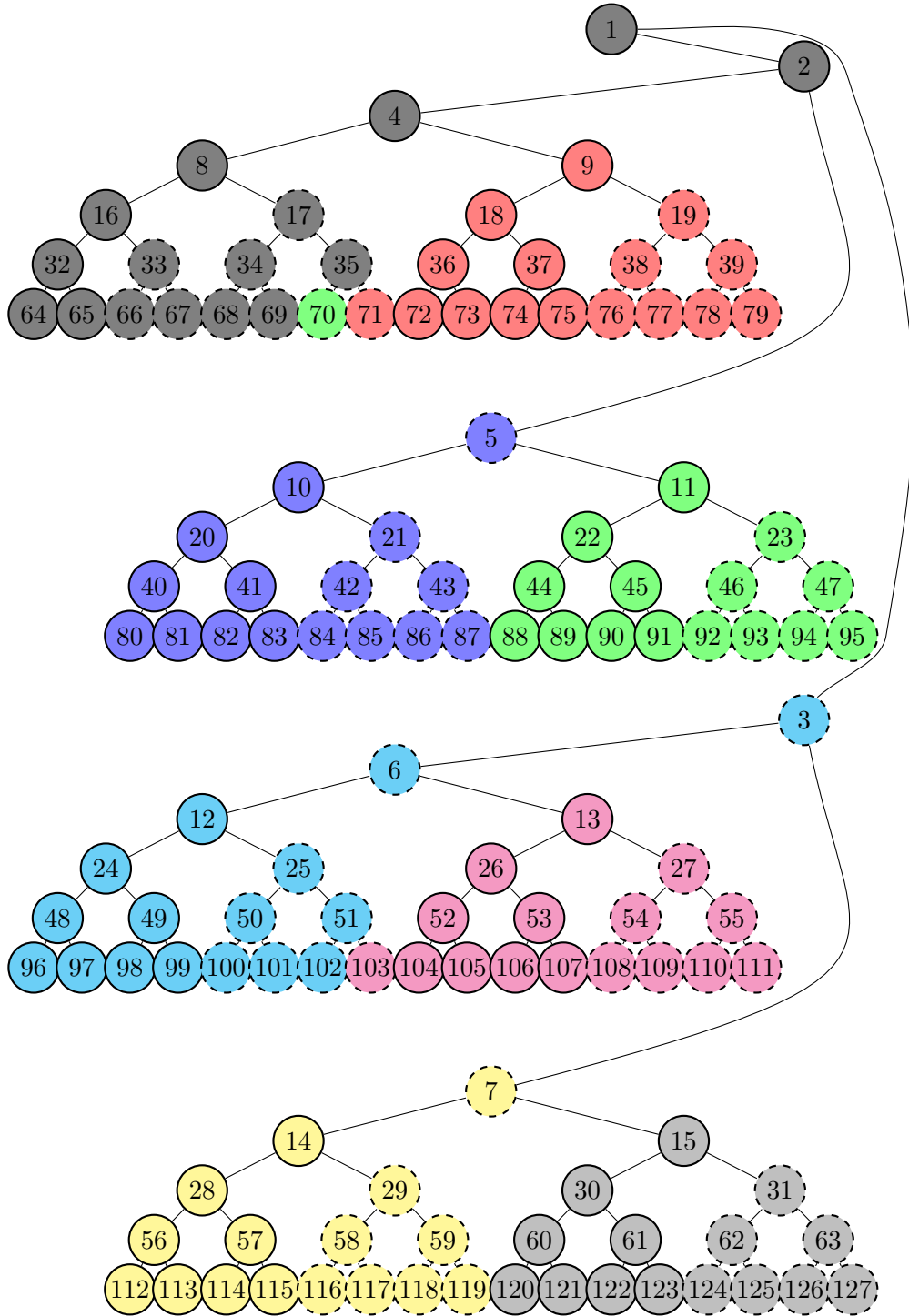


Figure 8: Guest graph $G = (V, E)$ (binary regular tree of height $h_G = 6$). The colours are related to the arrangement ϕ depicted in Figures 9 and 10.

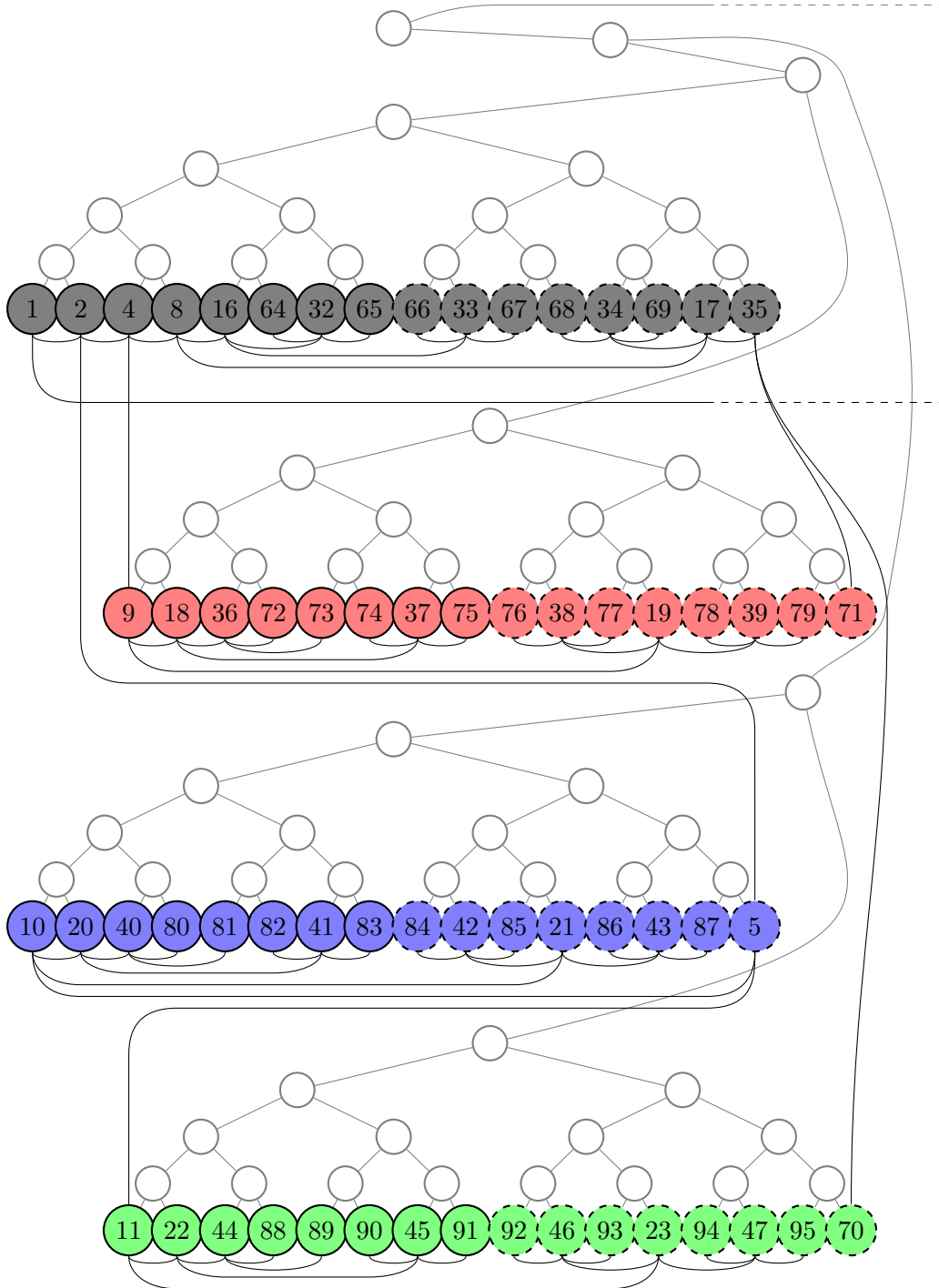


Figure 9: Arrangement ϕ for the guest graph $G = (V, E)$ depicted in Figure 8 – first part. Its objective function value is $OV(G, 2, \phi_A) = 584$.

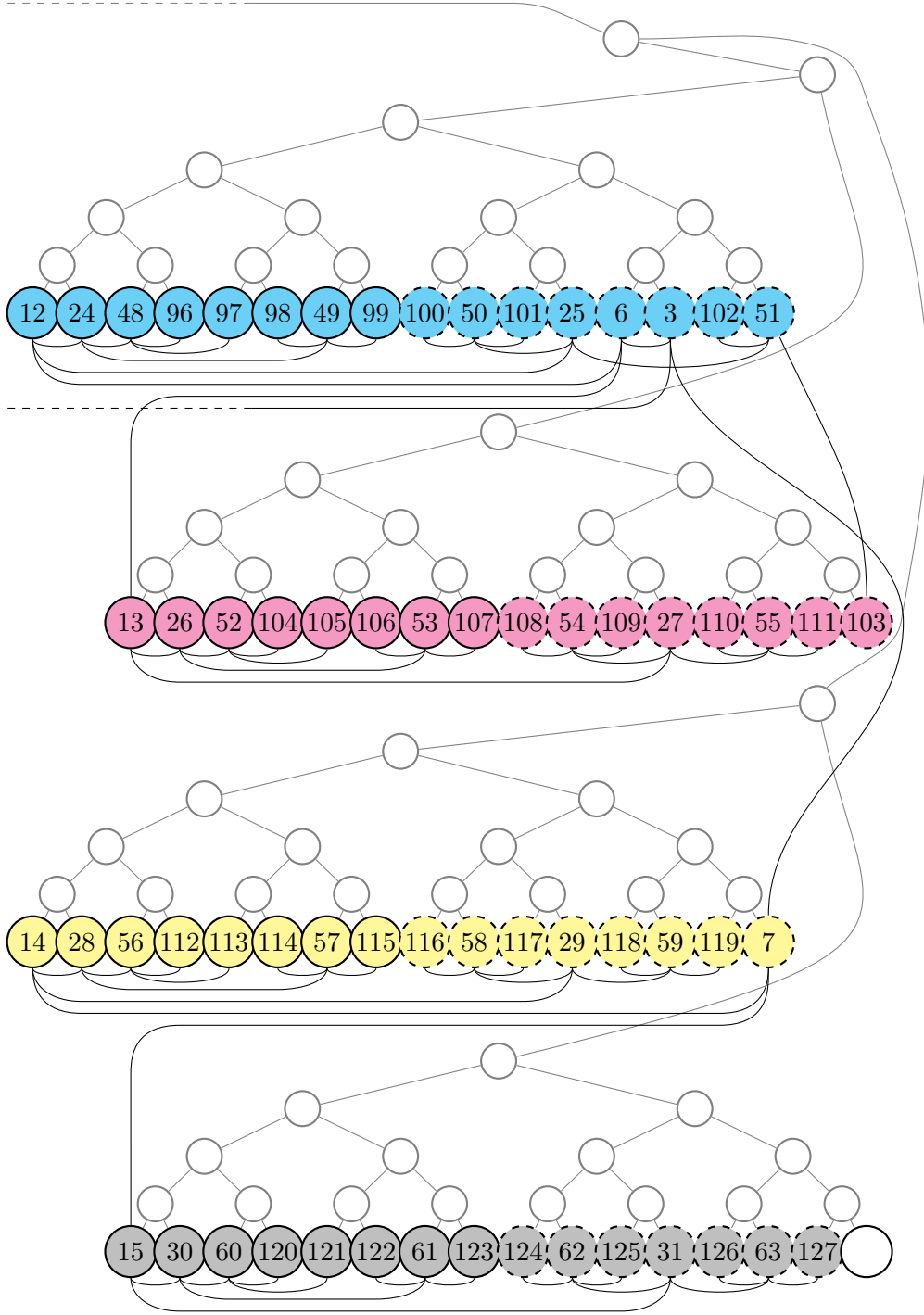


Figure 10: Arrangement ϕ for the guest graph $G = (V, E)$ depicted in Figure 8 – second part. Its objective function value is $OV(G, 2, \phi_A) = 584$.

The rest of this section is structured as follows: In Section 4.1 we introduce an algorithm to construct an optimal $2^{k'}$ -balanced partition \mathcal{V}^* in a regular binary tree. The optimality is proven in Section 4.2.

Section 4.3 provides a lower bound on the optimal value $c(G, k, \mathcal{V}^*)$ of the objective function of the $2^{k'}$ -BPP in a regular binary tree.

4.1 A solution algorithm for the $2^{k'}$ -BPP on regular binary trees

The algorithm consists of three simple steps. Let $t := h - k' + 2$ and $e := \lfloor \frac{h+1}{t} \rfloor - 1$.

1. First, we partition the tree G by cutting all edges $(u, v) \in E$ with $\text{level}(u) = h - it + 1$ and $\text{level}(v) = h - it + 1$, where $1 \leq i \leq e$. Roughly spoken, we separate e horizontal bands of height $t - 1$ from the input tree G , from the bottom to the top. The height of the remaining top part is then \hat{h} with $t - 1 \leq \hat{h} \leq 2t - 2$. Let p be the number of binary regular trees of height $t - 1$ contained in these bands.
2. Next consider the binary regular trees contained in the above mentioned bands and cut all edges connecting their roots with their right children, respectively. After that each root remains connected to the corresponding left basic subtree, thus forming a big partition set, since each root and its left basic subtree tree have $2^{t-2+1} - 1 + 1 = 2^{h-k'+1} = n_b$ vertices altogether.

On the other hand each of the right basic subtrees mentioned above has $2^{t-2+1} - 1 = 2^{h-k'+1} - 1 = n_b - 1$ vertices and needs one more vertex in order to form a big partition set. Let $q := \frac{2^{h-k'+1}-1}{2^{h-k'+1}}p$. We split $p - q$ of the right basic subtrees into isolated vertices, thus obtaining $(p - q)(2^{h-k'+1} - 1) = \left(p - \frac{2^{h-k'+1}-1}{2^{h-k'+1}}p\right)(2^{h-k'+1} - 1) = q$ isolated vertices. Each of them is paired with the remaining q non-split right basic subtrees in order to obtain further big partition sets. It is not difficult to check that $q \in \mathbb{N}$.

3. Finally, let us consider the top part consisting of a binary regular tree of height \hat{h} , $t - 1 \leq \hat{h} \leq 2t - 2$. We cut all edges $(u, v) \in E$ with $\text{level}(u) = h - (e + 1)t + 1$ and v being the right child of u . Analogously as above we obtain one big partition set for every vertex $u \in V$ with $\text{level}(u) = h - (e + 1)t + 1$ together with its corresponding left basic subtree. Moreover, each of the remaining right basic subtrees of the vertices u as above can be paired with one of the vertices $u' \in V$ with $\text{level}(u') < h - (e + 1)t + 1$, (roughly spoken, these are the vertices lying on the very top of the tree) in order to obtain further big partition sets. Again simple computations show that the number of the right basic subtrees mentioned above exceeds the number of the remaining vertices by exactly one. Hence just one right basic subtree of one vertex $u \in V$ with $\text{level}(u) = h - (e + 1)t + 1$ remains unpaired; this subtree builds the small partition set.

The following example illustrates this algorithm.

Example 14. Let us consider a binary regular tree $G = (V, E)$ of height $h = 5$ depicted in Figure 11 and let $k = 2^4 = 16$, i.e. $k' = 4$.

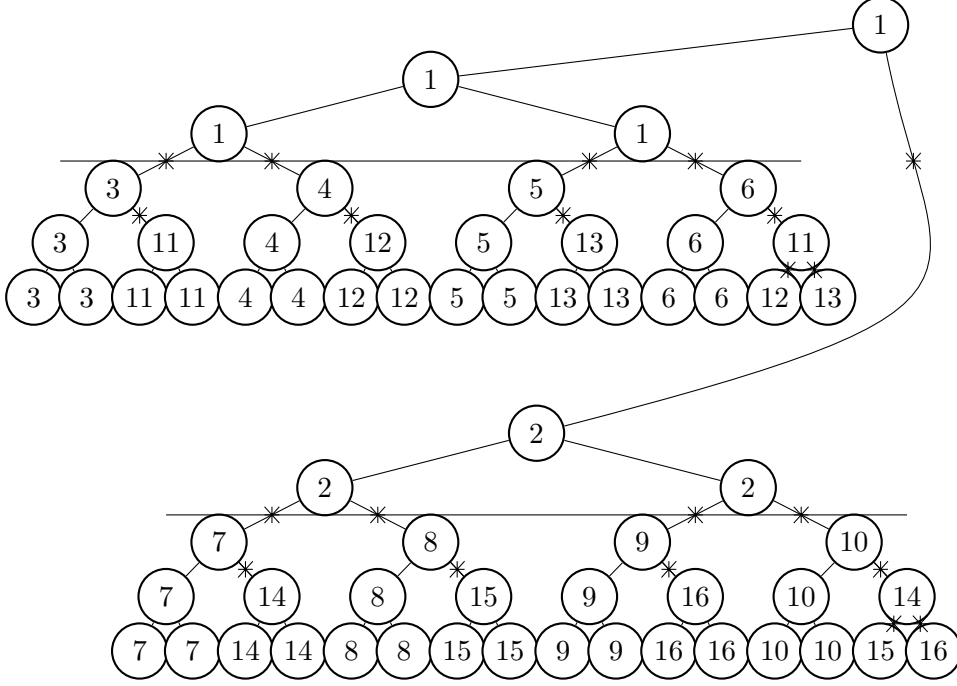


Figure 11: 16-balanced partition \mathcal{V}^* . Its objective value is $c(G, \mathcal{V}^*) = 21$. The numbers on the vertices indicate the index of the partition set to which the corresponding vertex belongs.

We have $t = h - k' + 2 = 5 - 4 + 2 = 3$ and $e = \lfloor \frac{h+1}{t} \rfloor - 1 = \lfloor \frac{5+1}{3} \rfloor - 1 = 1$.

1. Thus $i = 1$ and we cut all edges $(u, v) \in E$ with $\text{level}(u) = h - it = 5 - 1 \cdot 3 = 2$ and $\text{level}(v) = h - it + 1 = 5 - 1 \cdot 3 + 1 = 3$, i.e. the edges cut by the two horizontal lines in Figure 11.

2. Now, consider the bottom band consisting of $p = 8$ binary regular trees of height $t - 1 = 3 - 1 = 2$ each and in each of them cut all edges connecting the root with the right child. Each root connected to its corresponding left child forms a big partition set; we obtain the big partition sets V_i , $3 \leq i \leq 10$ depicted in Figure 11. Notice that in Figure 11 a partition set V_i contains the vertices marked by i , $1 \leq i \leq 16$.

Set $q := \frac{2^{h-k'+1}-1}{2^{h-k'+1}-1}p = \frac{2^{5-4+1}-1}{2^{5-4+1}-1}8 = 6$, and cut all edges of $p - q = 8 - 6 = 2$ arbitrarily chosen right basic subtrees. Pair each of the thereby arising isolated vertices with the remaining $q = 6$ right basic subtrees to obtain the big partition sets V_i $11 \leq i \leq 16$.

3. Finally, notice that $h - (e + 1)t + 1 = 5 - (1 + 1)3 + 1 = 0$. Thus we cut the edge connecting the root v_1 (note that $\text{level}(v_1) = 0$) with its right child according to the third step of the algorithm. We obtain the big partition sets V_1 and the small partition set V_2 depicted in Figure 11.

The objective function value of this 16-balanced partition $\mathcal{V}^* = \{V_1, V_2, \dots, V_{16}\}$, i.e. the number of the cut edges, equals $c(G, \mathcal{V}^*) = 21$. By applying the results of the following section we conclude that this is the optimal 16-balanced partition of G .

4.2 Proof of the optimality for the algorithm described in Section 4.1

The optimality proof will make use of the following reformulation of the k -BPP.

Consider the input graph $G = (V, E)$ of the k -BPP, a k -balanced partition $\mathcal{V} = V_1, V_2, \dots, V_k$, and the respective induced subgraphs $G[V_i]$, $1 \leq i \leq k$. Assume that $G[V_i]$ has l_i connected components $G_{i,j} = (V_{i,j}, E_{i,j})$ for $1 \leq j \leq l_i$, for $1 \leq i \leq k$. Define a new graph $G' = (V', E')$ which contains one representative vertex $\bar{v}_{i,j}$ for each connected component $G_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq l_i$. Two vertices \bar{v}_{i_1, j_1} and \bar{v}_{i_2, j_2} are connected in G' iff the connected components G_{i_1, j_1} , G_{i_2, j_2} are connected by an edge in G . Observe that if G is a tree, then G' is also a tree and the following equality holds

$$c(G, \mathcal{V}) = |E'| = |V'| - 1. \quad (12)$$

Thus the value of the objective function of the k -BPP corresponding to a k -balanced partition \mathcal{V} of a tree G equals the overall number of the connected components of the subgraphs induced in G by the partition sets of \mathcal{V} minus 1. Hence the goal of the k -BPP can be rephrased as follows: Determine a k -balanced partition \mathcal{V}^* such that the number of the connected components of the subgraphs induced in G by the partition sets is minimised.

Example 15. Let us consider the graph and the partition depicted in Figure 11. The tree G' is depicted in Figure 12.

The partition sets V_i , $1 \leq i \leq 10$, generate one connected component each, the partition sets V_i , $11 \leq i \leq 16$ generate two connected components each. Notice that equality (12) is fulfilled: the tree G' has 22 vertices and 21 edges and $c(G, \mathcal{V}) = 21$, see Example 14.

Denote by $n_i(\mathcal{V})$ be the number of partition sets in \mathcal{V} which induce i connected components each in G , for $i \in \mathbb{N}$. Notice now that the following observation holds.

Observation 16. Consider the $2^{k'}$ -BPP for a regular binary tree G of height h , $h \geq k'$, and let \mathcal{V}^* be the k -balanced partition computed by the algorithm described in Section 4.1. Then $n_1(\mathcal{V}^*) \geq n_1(\mathcal{V})$ implies $c(G, \mathcal{V}^*) \leq c(G, \mathcal{V})$, for any k -balanced partition \mathcal{V} of G .

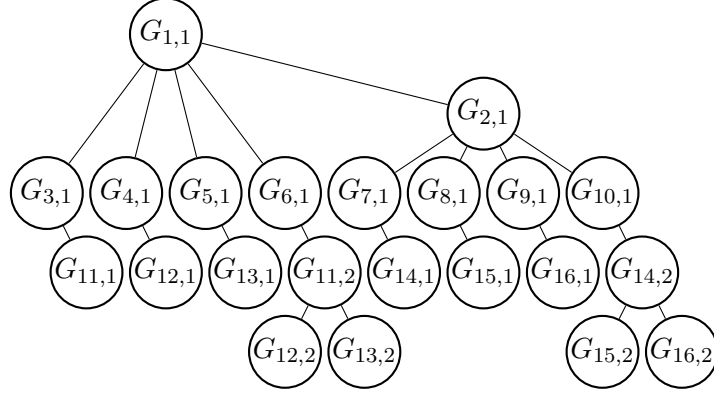


Figure 12: The graph G' corresponding to the regular binary G and the 16-partition \mathcal{V} depicted in Figure 11.

Proof. As argued above for every k -balanced partition \mathcal{V} the value $c(G, \mathcal{V})$ of the objective function equals the overall number of connected componets induced in G by the partition sets of \mathcal{V} minus 1, and hence $c(G, \mathcal{V}) = \sum_{i \in \mathbb{N}} in_i(\mathcal{V}) - 1$ holds. If $n_1(\mathcal{V}^*) \geq n_1(\mathcal{V})$ and since $n_i(\mathcal{V}^*) = 0$ for $i \geq 3$, the following equalities and inequalities hold:

$$c(G, \mathcal{V}) = \sum_{i \in \mathbb{N}} in_i(\mathcal{V}) - 1 \geq n_1(\mathcal{V}) + 2(k - n_1(\mathcal{V})) - 1 = 2k - n_1(\mathcal{V}) - 1 \geq$$

$$2k - n_1(\mathcal{V}^*) - 1 = n_1(\mathcal{V}^*) + 2(k - n_1(\mathcal{V}^*)) - 1 = n_1(\mathcal{V}^*) + 2n_2(\mathcal{V}^*) - 1 = c(G, \mathcal{V}^*). \quad (13)$$

□

Theorem 17. Let $G = (V, E)$ be a binary regular tree of height $h \geq 1$ and let $k = 2^{k'}$, where $1 \leq k' \leq h$. Then the algorithm presented in Section 4.1 yields an optimal k -balanced partition \mathcal{V}^* .

Proof. Due to Observation 16 it is enough to show that $n_1(\mathcal{V}^*) \geq n_1(\mathcal{V})$ for any k -balanced partition \mathcal{V} of G .

Let us first show that every k -balanced partition $\mathcal{V} =: \mathcal{V}_0$ can be transformed step by step into a sequence $\mathcal{V} =: \mathcal{V}_0, \mathcal{V}_1, \dots, \mathcal{V}_l = \mathcal{V}'$, $l \in \mathbb{N}$, of k -balanced partitions with the following properties:

- (a) $n_1(\mathcal{V}_t) \geq n_1(\mathcal{V}_{t-1})$ holds for every $t \in \{1, 2, \dots, l\}$, and
- (b) the big partition sets of \mathcal{V}' which induce one connected component in G each coincide with the big partition sets of \mathcal{V}^* which induce one connected component in G each.

In the following the steps of this transformation are explained. Consider a k -balanced partition \mathcal{V} whose big partition sets which induce one connected component in G each do not coincide with the corresponding partition sets of \mathcal{V}^* .

For every vertex $v \in V$ let $P_{\mathcal{V}}(v)$ be the uniquely determined partition set in \mathcal{V} such that $v \in P_{\mathcal{V}}(v)$. Consider the vertices $v \in V$ with $\text{level}(v) = h - it + 1$, where $1 \leq i \leq e$, $e = \lfloor \frac{h+1}{t} \rfloor - 1$, and choose among them a vertex with the largest level such that $P_{\mathcal{V}}(v) \neq P_{\mathcal{V}^*}(v)$. Perform now the following transformation steps.

Case 1. If the partition set $P_{\mathcal{V}}(v)$ consists of the vertex $v \in V$ together with the regular binary subtree of height $h - k'$ rooted at its right child, then exchange the subtrees rooted at the left and the right child of v , respectively, to obtain a new k -balanced partition \mathcal{V}' for which obviously $n_1(\mathcal{V}) = n_1(\mathcal{V}')$ holds.

Case 2. If Case 1 does not arise, then $P_{\mathcal{V}}(v)$ contains neither the regular binary subtree of height $h - k'$ rooted at the right child of v nor the regular binary subtree of height $h - k'$ rooted at its left child. At least one of these two subtrees does not build a small partition set in \mathcal{V} . Let this be the left subtree (otherwise we would apply an exchange of the two subtrees as in Case 1). Denote this subtree by T . If $P_{\mathcal{V}}(v)$ is a big component, then exchange the vertices contained in $P_{\mathcal{V}}(v)$ and the vertices of T (the one by one assignment of the corresponding vertices is done arbitrarily). Denote the resulting balanced partition by \mathcal{V}' . Clearly $P_{\mathcal{V}'}(v)$ induces one connected component in G . Moreover $P_{\mathcal{V}'}(u)$ induces more than one connected component in G , for every $u \in V(T)$, because T does not build a small partition set in \mathcal{V} . Then $n_1(\mathcal{V}') \geq n_1(\mathcal{V})$ holds (no partition sets inducing one connected component in G are destroyed). If $P_{\mathcal{V}}(v)$ is the small component, then again exchange the vertices contained in $P_{\mathcal{V}}(v)$ against all but one of the vertices in the subtree T of height $h - k'$ rooted at the left child of v (the one by one assignment of the corresponding vertices is again done arbitrarily). Then add the remaining vertex of T to the partition set containing v and resulting after that exchange. Denote the resulting balanced partition by \mathcal{V}' . Clearly $P_{\mathcal{V}'}(v)$ induces one connected component in G . Further $P_{\mathcal{V}'}(u)$ induces more than one connected component in G , for every $u \in V(T)$, because T does not build a small partition set in \mathcal{V} . Thus $n_1(\mathcal{V}') \geq n_1(\mathcal{V})$ holds again for the same reason as above.

We repeat the above transformation step as long as there are vertices v for which $P_{\mathcal{V}}(v) \neq P_{\mathcal{V}^*}(v)$, where \mathcal{V} denotes the most recently constructed k -balanced partition. We end up with a k -balanced partition \mathcal{V}' which contains as partition sets all big partition sets of \mathcal{V}^* which induce one connected component in G , respectively. Moreover \mathcal{V}' fulfills the following inequality

$$n_1(\mathcal{V}') \geq n_1(\mathcal{V}). \quad (14)$$

Notice finally that by removing from G all big partition sets of \mathcal{V}^* which induce one connected component in G , respectively, we obtain a graph whose largest connected component contains at most $2^{h-k'} - 1$ vertices. So, if \mathcal{V}' contains any partition sets which induce one connected component in G besides the big partition sets of \mathcal{V}^* inducing one connected component, then these partition sets should be small ones. Finally, since in

every k -balanced partition there is only one small partition set and the small partition set in \mathcal{V}^* induces one connected component in G we get

$$n_1(\mathcal{V}^*) \geq n_1(\mathcal{V}'). \quad (15)$$

By combining (14) and (15) we get $n_1(\mathcal{V}^*) \geq n_1(\mathcal{V})$ and this completes the proof. \square

4.3 The optimal value of the $2^{k'}$ -BPP on regular binary trees

According to equality (13) the optimal value $c(G, \mathcal{V}^*)$ of the $2^{k'}$ -BPP on a regular binary tree of height h for $1 \leq k' \leq h$ is given as $c(G, \mathcal{V}^*) = 2k - n_1(\mathcal{V}^*) - 1$. Recall that $n_1(\mathcal{V}^*)$ is the number of partitions sets of \mathcal{V}^* which induce exactly one connected component each in G . Observe that for every $i \in \mathbb{N}$, $1 \leq i \leq e + 1$, the algorithm presented in Section 4.1 constructs exactly 2^{h-it+1} big partition sets inducing one connected component in G , respectively. More precisely, it constructs one such partition set for each vertex of level $h - it + 1$, where the partition sets arise in the i -th bands of height $t - 1$ by cutting the edge joining the above mentioned vertices to their right children, respectively. Finally in its last step the algorithm constructs the small partition set which also induces one connected component in G .

Thus the following equality holds

$$n_1(\mathcal{V}^*) = 1 + \sum_{i=1}^{e+1} 2^{h-it+1} = 1 + 2^{h+1} \frac{1 - \left(\frac{1}{2^t}\right)^{e+1}}{2^t \left(1 - \frac{1}{2^t}\right)}, \quad (16)$$

and this implies

$$c(G, \mathcal{V}^*) = 2k - 2 - 2^{h+1} \frac{1 - \left(\frac{1}{2^t}\right)^{e+1}}{2^t \left(1 - \frac{1}{2^t}\right)}. \quad (17)$$

For technical reasons we derive a lower bound for $c(G, \mathcal{V}^*)$ which is given as a closed formula depending just on k .

Lemma 18. *Let $G = (V, E)$ be a regular binary tree of height $h \geq 1$ and let $k = 2^{k'}$, where $1 \leq k' \leq h$. Let \mathcal{V}^* be the optimal k -balanced partition in G computed by the algorithm described in Section 4.1. The optimal value $c(G, \mathcal{V}^*)$ of the k -BPP in G fulfills the following (in)equalities:*

$$\begin{aligned} c(G, \mathcal{V}^*) &\geq \frac{10}{7}k - 2, \text{ if } k' \leq h - 1, \\ c(G, \mathcal{V}^*) &= \frac{4}{3}k - \frac{3}{2} + \frac{1}{6}(-1)^{\log_2 k}, \text{ if } k' = h, \text{ and} \\ c(G, \mathcal{V}^*) &= \frac{3}{2}k - 2, \text{ if } k' \leq \lfloor \frac{h}{2} \rfloor + 1. \end{aligned}$$

Proof. If $k' \leq h - 1$ we get $t = h - k' + 2 \geq h - (h - 1) + 2 = 3$ and the following inequalities hold

$$n_1(\mathcal{V}^*) \leq 1 + 2^{h+1} \left(\frac{1}{1 - \frac{1}{2^t}} - 1 \right) = 1 + \frac{2^{h+1} \frac{1}{2^t}}{1 - \frac{1}{2^t}} = 1 + \frac{2^{h+1-t}}{1 - \frac{1}{2^t}} = 1 + \frac{2^{k'-1}}{1 - \frac{1}{8}} =$$

$$1 + \frac{2^{k'}}{\frac{7}{4}} = 1 + \frac{4}{7}k.$$

The last inequality implies

$$(G, \mathcal{V}^*) = 2k - n_1(\mathcal{V}^*) - 1 \geq \frac{10}{7}k - 2. \quad (18)$$

If $k' = h$ we get $t = h - k' + 2 = 2$ and $e = \lfloor \frac{h+1}{t} \rfloor - 1 = \lfloor \frac{h+1}{2} \rfloor - 1$. If h is odd, $e = \frac{h+1}{2} - 1$ holds, and if h is even we get $e = \lceil \frac{h+1}{2} \rceil = \frac{h}{2}$. By setting this values for e and t in equation (16) and simplifying we get

$$n_1(\mathcal{V}^*) = \begin{cases} \frac{2}{3}2^h + \frac{2}{3} & \text{if } h \text{ is odd} \\ \frac{2}{3}2^h + \frac{1}{3} & \text{if } h \text{ is even} \end{cases}$$

By plugging these expressions for $n_1(\mathcal{V}^*)$ into equation 17 we obtain

$$c(G, \mathcal{V}^*) = 2k - n_1(\mathcal{V}^*) - 1 = \frac{4}{3}k - \frac{3}{2} + \frac{1}{6}(-1)^{\log_2 k} \text{ in both cases.} \quad (19)$$

If $k' \leq \lfloor \frac{h}{2} \rfloor + 1$ we get $t = h - k' + 2$ and $e = \lfloor \frac{h+1}{t} \rfloor - 1 = \left\lfloor \frac{h+1}{h-k'+2} \right\rfloor - 1 \leq \left\lfloor \frac{h+1}{h - \lfloor \frac{h}{2} \rfloor + 1} \right\rfloor - 1$. By distinguishing the two cases when h is odd and h is even it can be easily observed that $e = 0$ in both cases. By substituting e by 0 in equation 16 we get

$$n_1(\mathcal{V}^*) = 1 + 2^{h+1} \frac{1}{2^t} = 1 + \frac{2^{h+1}}{2^{h-k'+2}} = 1 + 2^{k'-1} = 1 + \frac{2^{k'}}{2} = 1 + \frac{k}{2},$$

and then by plugging this into equation 17

$$c(G, \mathcal{V}^*) = 2k - 1 - \frac{k}{2} - 1 = \frac{3}{2}k - 2. \quad (20)$$

□

5 The approximation ratio of Algorithm 1

In order to estimate an approximation ratio ρ for Algorithm 1 we will exploit the relationship between the *DAPT* on binary regular trees and the *k-BPP* and obtain a lower bound for the objective function value of the *DAPT* in terms of the special case of *k-BPP* where k is a power of two.

Let the guest graph $G = (V, E)$ and the host graph T be regular binary trees of heights $h_G \geq 1$ and $h = h_G + 1$, respectively, and let ϕ be an arbitrary arrangement with corresponding objective value $OV(G, 2, \phi) = \sum_{(u,v) \in E} d_T(\phi(u), \phi(v))$. The length $d_T(\phi(u), \phi(v))$ of the unique $\phi(u)$ - $\phi(v)$ -path in the binary regular tree T is even for any two vertices u and v in G (see and Observation 6). Moreover, $2 \leq d_T(\phi(u), \phi(v)) \leq 2h$ holds for any two vertices u and v in G . Let $a_i(\phi)$, $1 \leq i \leq h$, be the number of edges which contribute to the value of the objective function by $2i$, i.e. the number of edges $(u, v) \in E(G)$ for which $d_T(\phi(u), \phi(v)) = 2i$ holds. Then

$$OV(G, 2, \phi) = a_h(\phi) \cdot 2h + a_{h-1}(\phi) \cdot 2(h-1) + \dots + a_1(\phi) \cdot 2 = 2 \sum_{i=1}^h a_i(\phi) i. \quad (21)$$

The number of edges which contribute to the value of the objective function by *at least* $2i$, i.e. the number of edges $(u, v) \in E(G)$ for which $d_T(\phi(u), \phi(v)) \geq 2i$ holds, is given by the following the **partial sums**

$$s_i(\phi) := \sum_{j=i}^h a_j(\phi) \text{ for all } 1 \leq i \leq h. \quad (22)$$

Clearly, the following equalities hold

$$a_i(\phi) = \begin{cases} s_i(\phi) - s_{i+1}(\phi) & \text{for } 1 \leq i \leq h-1 \\ s_i(\phi) & \text{for } i = h \end{cases}. \quad (23)$$

By plugging this into the objective function in (21) we get

$$OV(G, 2, \phi) = 2 \sum_{i=1}^h a_i(\phi) i = 2 \left(\left(\sum_{i=1}^{h-1} i(s_i(\phi) - s_{i+1}(\phi)) \right) + h s_h(\phi) \right) = 2 \sum_{i=1}^h s_i(\phi). \quad (24)$$

Example 19. *Let us consider the guest graph in Figure 2 and the arrangement ϕ_A obtained by applying Algorithm 1; this arrangement is depicted in Figure 4. The coefficients $a_i(\phi_A)$, $s_i(\phi_A)$, for $1 \leq i \leq h$ are listed in Table 1.*

By applying (21) and (24) we obtain the corresponding objective function value, respectively, as follows:

$$OV(G, 2, \phi_A) = 2 \sum_{i=1}^h a_i(\phi_A) i = 2(5 \cdot 1 + 5 \cdot 2 + 3 \cdot 3 + 1 \cdot 4) = 56, \quad (25)$$

| i | 4 | 3 | 2 | 1 |
|---------------|---|---|---|----|
| $a_i(\phi_A)$ | 1 | 3 | 5 | 5 |
| $s_i(\phi_A)$ | 1 | 4 | 9 | 14 |

Table 1: Coefficients $a_i(\phi_A)$ and partial sums $s_i(\phi_A)$, for $1 \leq i \leq h$.

$$OV(G, 2, \phi_A) = 2 \sum_{i=1}^h s_i(\phi_A) = 2(14 + 9 + 4 + 1) = 56. \quad (26)$$

For $h_G = 0$ the arrangement ϕ_A is obviously optimal, so let us assume that $h_G \geq 1$ through the rest of this section. The next lemma and its corollary give closed formulas for the coefficients $a_i(\phi_A)$ and the partial sums $s_i(\phi_A)$, $1 \leq i \leq h$, corresponding to the arrangement computed by Algorithm 1.

Lemma 20. *Let the guest graph $G = (V, E)$ and the host graph T be binary regular trees of heights $h_G \geq 1$ and $h = h_G + 1$, respectively, and let ϕ_A be the arrangement computed by Algorithm 1. Then the coefficients $a_i(\phi_A)$, $1 \leq i \leq h$, are given as follows:*

$$a_i(\phi_A) = \begin{cases} \frac{2}{3}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G} & \text{for } i = 1 \\ \frac{7}{12}2^{h_G} + \frac{1}{2} + \frac{1}{6}(-1)^{h_G} & \text{for } i = 2 \\ 3 \cdot 2^{h_G-i} & \text{for } 3 \leq i < h \\ 1 & \text{for } i = h \end{cases} \quad (27)$$

Proof. Consider first $i = 1$ and determine the number of edges contributing to the objective value $OV(G, 2, \phi_A)$ by exactly 2. Let us first neglect the pair-exchanges done in pseudocode line 9. Then there are only two possibilities how to arrange an edge in Algorithm 1 in such a way that it contributes by 2 to the objective value

1. Either it is taken over from the recursive arrangements in pseudocode line 6
2. or it is produced by placing the root v_1 in pseudocode line 7.

In the latter case the following property P must hold: (P) A child of the root v_1 and v_1 itself are placed to children vertices of a common father in T . Since the children of the root v_1 are roots in the previous recursion step, and since the roots are always placed on the middle leaf, $h_G = 1$ has to hold in the corresponding recursive run, i.e. in the run when property P is fulfilled. So exactly one edge contributing by 2 to the value of the objective function arises in every such recursive run with $h_G = 1$ (see also Figures 15 and 16 in Appendix). There are 2^{h_G-1} such runs, one for each vertex with level $h_G - 1$ (playing the role of the root). Thus, if the pair exchange step is neglected, there are 2^{h_G-1} edges which contribute 1 to the value of the objective function.

Let $pe(h_G)$ be the number of pair-exchanges done in pseudocode line 9 when applying the algorithm on a guest graph of height h_G . We prove that

$$pe(h_G) = \frac{1}{6}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G} \quad (28)$$

by induction on the height h_G . For $h_G = 1$ the formula in (28) yields $pe(h_G) = 0$ which is obviously correct (see also Figures 15 and 16 in Appendix). Analogous arguments as in the proof of Lemma 10 show that the following recursive equations hold for $h_G \geq 1$:

$$pe(h_G + 1) = \begin{cases} 2pe(h_G) + 1 & \text{for } h_G + 1 \text{ odd} \\ 2pe(h_G) & \text{for } h_G + 1 \text{ even} \end{cases} \quad (29)$$

So by applying (28) and (29) we get:

$$pe(h_G + 1) = 2pe(h_G) + 1 = 2 \left(\frac{1}{6}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G} \right) + 1 =$$

$$\frac{1}{6}2^{h_G+1} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G},$$

if h_G is odd, and

$$pe(h_G + 1) = 2pe(h_G) = 2 \left(\frac{1}{6}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G} \right) = \frac{1}{6}2^{h_G+1} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G},$$

if h_G is even. Thus also $pe(h_G + 1)$ fulfills (28), which completes the inductive proof of (28).

In Lemma 9 it was proven that every pair-exchange done in pseudocode line 9 increases by 1 the number of edges contributing by 2 to the value of the objective function. Thus we get

$$a_1(\phi_A) = 2^{h_G-1} + pe(h_G) = \frac{2}{3}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G},$$

and hence the claim of the lemma holds for $i = 1$.

Let $i = 2$. We use the same technique: We count vertices with $\text{level}(v) = h_G - 1$ and the number of vertices with $\text{level}(v) = h_G - 2$ first. Edges which contribute by 4 to the value of the objective function arise only in recursive runs where the guest graph has height 1 or 2 and is rooted at a vertex with level $h_G - 1$ or $h_G - 2$, respectively. In every such recursive run exactly one edge of that kind arises, see also Figures 15, 16, 17 and 18). Then we consider the effect of the pair-exchanges done in pseudocode line 9. According to Lemma 9 each such pair-exchange reduces by one the number of edges which contribute to the value of the objective function by 4, so we get

$$a_2(\phi_A) = 2^{h_G-1} + 2^{h_G-2} - pe(h_G) = \frac{7}{12}2^{h_G} + \frac{1}{2} + \frac{1}{6}(-1)^{h_G},$$

and hence the claim of the lemma holds for $i = 2$.

Consider now the case $i \geq 3$. According to Lemma 9 the pair-exchanges done in pseudocode line 9 have no effect on the number of edges of G which contribute to the

value of the objective function by $2i$, if $i \geq 3$. So for $3 \leq i < h$ the pair-exchanges can be neglected and we get

$$a_i(\phi_A) = 2^{h_G-(i-1)} + 2^{h_G-i} = 3 \cdot 2^{h_G-i},$$

in compliance with the claim of the lemma.

Finally, for $i = h$ there is exactly one edge which contributes by $2h$ to the value of the objective function, namely the one joining the root of G with his right child. \square

Corollary 21. *Let the guest graph $G = (V, E)$ and the host graph T be binary regular trees of heights $h_G \geq 1$ and $h = h_G + 1$, respectively, and let ϕ_A be the arrangement computed by Algorithm 1. Then the coefficients $s_i(\phi_A)$, $1 \leq i \leq h$, are given as follows:*

$$s_i(\phi_A) = \begin{cases} 2 \cdot 2^{h_G} - 2 & \text{for } i = 1 \\ \frac{4}{3}2^{h_G} - \frac{3}{2} + \frac{1}{6}(-1)^{h_G} & \text{for } i = 2 \\ 6 \cdot 2^{h_G-i} - 2 & \text{for } 3 \leq i \leq h \end{cases} \quad (30)$$

Proof. This corollary is a straightforward consequence of the definition of $s_i(\phi)$, $1 \leq i \leq h$, (see (22)) and of Lemma 20.

- For $i = h$ we get: $s_h(\phi_A) = a_h(\phi_A) = 1 = 6 \cdot 2^{h_G-(h_G+1)} - 2 = 6 \cdot 2^{h_G-i} - 2$.
- For $3 \leq i < h$ we get by induction on i and starting with $i = h$: $s_i(\phi_A) = a_i(\phi_A) + s_{i+1}(\phi_A) = 3 \cdot 2^{h_G-i} + 6 \cdot 2^{h_G-(i+1)} - 2 = 6 \cdot 2^{h_G-i} - 2$.
- For $i = 2$ we get: $s_2(\phi_A) = a_2(\phi_A) + s_3(\phi_A) = \frac{7}{12}2^{h_G} + \frac{1}{2} + \frac{1}{6}(-1)^{h_G} + 6 \cdot 2^{h_G-3} - 2 = \frac{4}{3}2^{h_G} - \frac{3}{2} + \frac{1}{6}(-1)^{h_G}$.
- For $i = 1$ we get: $s_1(\phi_A) = a_1(\phi_A) + s_2(\phi_A) = \frac{2}{3}2^{h_G} - \frac{1}{2} - \frac{1}{6}(-1)^{h_G} + \frac{4}{3}2^{h_G} - \frac{3}{2} + \frac{1}{6}(-1)^{h_G} = 2 \cdot 2^{h_G} - 2$.

\square

Next we give a lower bound for $s_i(\phi)$, where ϕ is an arbitrary arrangement of the vertices of the guest graph G with height h_G into the leaves of the host graph T with height $h := h_G + 1$ and i is some integer between 1 and h . $s_i(\phi)$ is the number of edges of G which contribute by at least $2i$ to the value of the objective function. Obviously, each such edge joins vertices of G which are arranged at the leaves of *different binary regular subtrees* of T of height $i - 1$ and rooted at vertices of level $h - (i - 1) = h_G - i + 2 =: k'$. Clearly, there are $2^{k'}$ such subtrees of T . Let us denote them by $T_j^{(i)}$, for $1 \leq j \leq 2^{k'}$. Let $V_j^{(i)} \subset V(G)$ be the set of vertices of G which are arranged at the leaves of $T_j^{(i)}$. Since for all leaves b of T but one there is some vertex $v \in V(G)$ with $\phi(v) = b$, $|V_j^{(i)}| = 2^{i-1}$ holds for all but one index j , $1 \leq j \leq 2^{k'}$, and for the exception, say j_0 ,

$|V_{j_0}^{(i)}| = 2^{i-1} - 1$ holds. Thus $\mathcal{V}^{(i)} := \{V_j^{(i)} | 1 \leq j \leq 2^{k'}\}$ is k -balanced partition of G with $k = 2^{k'}$, $k' = h_G - i + 2$, and $s_i(\phi) = c(G, \mathcal{V}^{(i)})$. Let \mathcal{V}_i^* be the optimal k -balanced partition of G (which can be computed by the algorithm presented in Section 4.1 and for which lower bounds of the objective function value as in Section 4.3 are known). Then $s_i(\phi) \geq c(G, \mathcal{V}_i^*)$ holds for all i , $2 \leq i \leq h$ and for every arrangement ϕ . Let us denote the lower bounds $s_i^L := c(G, \mathcal{V}_i^*)$ for $2 \leq i \leq h$, and $s_1^L := |V(G)| - 1 = 2^h - 2$ for $i = 1$.

Thus we get

$$OV(G, 2, \phi) \geq 2 \sum_{i=1}^h s_i^L \text{ for all arrangements } \phi. \quad (31)$$

Notice that $s_1(\phi) = |E(G)| = |V(G)| - 1 = s_1^L$ holds for every arrangement ϕ of the guest graph G . Notice, moreover, that for $h_G \leq 4$ the bound in (31) is tight, in the sense that it matches the optimal value of the objective function of $DAPT(G, 2)$, as illustrated in the following tables.

| i | 2 | 1 |
|---------------|---|---|
| $s_i(\phi_A)$ | 1 | 2 |
| s_i^L | 1 | 2 |

Table 2: Partial sums $s_i(\phi_A)$ and the corresponding lower bounds s_i^L , where $1 \leq i \leq h$, for a guest graph $G = (V, E)$ of height $h_G = 1$.

| i | 3 | 2 | 1 |
|---------------|---|---|---|
| $s_i(\phi_A)$ | 1 | 4 | 6 |
| s_i^L | 1 | 4 | 6 |

Table 3: Partial sums $s_i(\phi_A)$ and the corresponding lower bounds s_i^L , where $1 \leq i \leq h$, for a guest graph $G = (V, E)$ of height $h_G = 2$.

| i | 4 | 3 | 2 | 1 |
|---------------|---|---|---|----|
| $s_i(\phi_A)$ | 1 | 4 | 9 | 14 |
| s_i^L | 1 | 4 | 9 | 14 |

Table 4: Partial sums $s_i(\phi_A)$ and the corresponding lower bounds s_i^L , where $1 \leq i \leq h$, for a guest graph $G = (V, E)$ of height $h_G = 3$.

| i | 5 | 4 | 3 | 2 | 1 |
|---------------|---|---|----|----|----|
| $s_i(\phi_A)$ | 1 | 4 | 10 | 20 | 30 |
| s_i^L | 1 | 4 | 10 | 20 | 30 |

Table 5: Partial sums $s_i(\phi_A)$ and the corresponding lower bounds s_i^L , where $1 \leq i \leq h$, for a guest graph $G = (V, E)$ of height $h_G = 4$.

In general, the lower bound in (31) is not tight. Already for $h_G = 5$ there is a gap between the value of the objective function corresponding to the arrangement ϕ_A generated by the algorithm \mathcal{A} and the lower bound, as shown in Table 6 below. Moreover, in the following example we prove that ϕ_A is an optimal arrangement for $h_G = 5$. Thus we conclude that the lower bound in (31) does not match the optimal value of the objective function of $DAPT(G, 2)$ already for $h_G = 5$.

Example 22. Consider the guest graph $G = (V, E)$ to be a complete binary tree of height $h_G = 5$ ordered according to the canonical ordering. The partial sums $s_i(\phi_A)$ and the lower bounds s_i^L , for $1 \leq i \leq h$, computed according to Corollary 21, inequality (31) and equation (17), are given in Table 6 below.

| i | 6 | 5 | 4 | 3 | 2 | 1 |
|---------------|---|---|----|----|----|----|
| $s_i(\phi_A)$ | 1 | 4 | 10 | 22 | 41 | 62 |
| s_i^L | 1 | 4 | 10 | 21 | 41 | 62 |

Table 6: Partial sums $s_i(\phi_A)$ and the corresponding lower bounds s_i^L for $1 \leq i \leq h$.

Thus in the case $h_G = 5$ the bound of inequality (31) does not match the objective function value corresponding to ϕ_A because $s_3(\phi_A) > s_3^L$. Now a natural question arises:

Question: Does the bound in (31) match the optimal value of the objective function of the $DAPT(G, 2)$ if $h_G = 5$?

We show that ϕ_A is an optimal arrangement if $h_G = 5$, and hence the answer to the above question is “no”. Indeed, assume that ϕ_A is not optimal and let ϕ_* be an optimal arrangement. Observe that $s_6(\phi_*) = 1$ has to hold because $s_6(\phi_*) \geq 2$ leads to a contradiction as follows:

$$OV(G, 2, \phi_*) - OV(G, 2, \phi_A) = 2 \sum_{i=1}^6 (s_i(\phi_*) - s_i(\phi_A)) =$$

$$2 \sum_{i=1}^5 (s_i(\phi_*) - s_i(\phi_A)) + 2(s_6(\phi_*) - s_6(\phi_A)) > 2 \sum_{i=1}^5 (s_i^L - s_i(\phi_A)) + 2 = -2 + 2 = 0.$$

By similar arguments we would get $s_i(\phi_*) = s_i^L = s_i(\phi_A)$ for $i \in \{1, 2, 4, 5\}$. Since $s_6(\phi_*) = 1$ there will only be one edge of G such that its endpoints are mapped to leaves of the right and the left basic subtrees of T , respectively. Clearly this edge can only be

(v_1, v_2) or (v_1, v_3) . Assume w.l.o.g. that this edge is (v_1, v_3) and that ϕ_* arranges the right basic subtree of G (of height 4) at the leaves of the right basic subtree T^r of T . Since algorithm \mathcal{A} yields an optimal arrangement for $h_G \leq 4$, we can then assume w.l.o.g. that ϕ_* and ϕ_A arrange the right basic subtree of G in the same way. Now consider the left basic subtree of G together with the root v_1 and denote this subgraph of G by G_1 . ϕ_* arranges G_1 at the leaves of the left basic subtree T^l of T . Let $G_1^{(a)}$ and $G_1^{(b)}$ the two subgraphs of G_1 arranged by ϕ_* at the leaves of the left and the right basic subtrees of T^l , denoted by T^{ll} and T^{lr} , respectively. Since the number of edges which contribute by at least $2 \cdot 5$ to the value $OV(G, 2, \phi_*)$ is $s_5(\phi_*) = 4$, there are just two edges e_i of G_1 such that ϕ_* arranges one endpoint of e_i to some leaf of T^{ll} and the other endpoint of e_i to some leaf of T^{lr} , for $i = 1, 2$. Recalling that $|V(G_1^{(a)})| = |V(G_1^{(b)})|$ we can easily convince ourselves (in the worst case by using complete enumeration) that one of the edges e_i , $i = 1, 2$, has to coincide with one of the two edges (v_1, v_2) or (v_2, v_5) (or (v_2, v_4) , symmetrically). We obtain two cases. (A) If $e_1 = (v_1, v_2)$, then $e_2 = (v_2, v_5)$ must hold. (B) Otherwise, if $e_1 = (v_2, v_5)$ and $e_2 \neq (v_1, v_2)$, then e_2 has to join some leave of the left basic subtree of $G_1 \setminus \{v_1\}$ to its father, e.g. $e_2 = (v_{19}, v_{39})$. The edges e_i , $i = 1, 2$, fully determine the corresponding subgraphs $G_1^{(a)}$ and $G_1^{(b)}$, each of them having 16 vertices. For every realisation of e_i , $i = 1, 2$, the problems $DAPT(G_1^{(a)}, 2)$ and $DAPT(G_1^{(b)}, 2)$ can be solved by complete enumeration to observe that the corresponding optimal values coincide with the values of the objective function corresponding to the arrangement of the respective subgraphs according to ϕ_A . Notice that it is enough to do the complete enumeration for the DAPTs resulting in the case A and for the DAPTs resulting in the case B with $e_2 = (v_{19}, v_{39})$; all other possible realisations of e_2 in case B lead to $G_1^{(a)}$, $G_1^{(b)}$ which are isomorphic to $G_1^{(a)}$, $G_1^{(b)}$ obtained for $e_2 = (v_{19}, v_{39})$, respectively.

Notice finally, that the answer “no” to the question posed above is not surprising. In general a collection of optimal 2^k -balanced partitions, $1 \leq k \leq h_G$, of a binary tree G of height h_G , does not need to coincide with the 2^k -balanced partitions of G defined in accordance with some feasible solution of the data arrangement problem in G . The reason is that the collection of 2^k -balanced partitions, $1 \leq k \leq h_G$, which arises in accordance with some arrangement ϕ , is laminar, meaning that the partition sets of the 2^k -balanced partition are obtained as particular partitions of the partition sets of the 2^{k-1} -balanced partition, for $2 \leq k \leq h_G$. More concretely, if $\mathcal{V} = \{V_1^{(1)}, V_1^{(2)}\}$ is the 2-balanced partition in the laminar collection of balanced partitions, then the 4-balanced partition is obtained by partitioning $V_1^{(1)}$ and $V_1^{(2)}$ into 2-balanced partitions each, and so on, until the partition sets are pairs of vertices, and hence a 2^{h_G} -balanced partition results. On the other side, in general, a collection of optimal 2^k -balanced partitions of a complete binary tree G of height h_G , for $1 \leq k \leq h_G$, is not necessarily laminar.

Now we can state the main result of this paper.

Theorem 23. *Let the guest graph $G = (V, E)$ and the host graph T be regular binary trees of heights $h_G \geq 1$ and $h = h_G + 1$, respectively. Then Algorithm 1 is a $\frac{203}{200}$ -approximation algorithm.*

Proof. The cases $h_G = 0$, $h_G = 1$, $h_G = 2$ are obvious: the arrangements ϕ_A obtained from Algorithm 1 are optimal in these cases, respectively, as one can convince himself by simple arguments or by full enumeration (see also Figures 13 to 18 in Appendix).

Let $h_G \geq 3$. Consider the difference $OV(G, 2, \phi) - 2 \sum_{i=1}^h s_i^L = 2 \sum_{i=1}^h s_i(\phi_A) - 2 \sum_{i=1}^h s_i^L$, and set

$$D := \sum_{i=1}^h (s_i(\phi_A) - s_i^L).$$

As mentioned above $s_h^L = 1 = s_h(\phi_A)$.

For $s_2^L = c(G, \mathcal{V}_2^*)$ we have $k' = h_G - 2 + 2 = h_G$ and $k = 2^{k'} = 2^{h_G}$, and thus we can apply Lemma 18 and Corollary 21 to obtain $s_2^L = \frac{4}{3}2^{h_G} - \frac{3}{2} + \frac{1}{6}(-1)^{h_G} = s_2(\phi_A)$.

Let us consider $s_i^L = c(G, \mathcal{V}_i^*)$, where $3 \leq i \leq h_G$. If $1 \leq k' \leq \left\lfloor \frac{h_G}{2} \right\rfloor + 1$, with $k' = h_G - i + 2$ we obtain the condition $i \geq h_G - \left\lfloor \frac{h_G}{2} \right\rfloor + 1$. For $h_G \geq 3$ the inequality $h_G - \left\lfloor \frac{h_G}{2} \right\rfloor + 1 \geq 3$ holds and hence Corollary 21 applies. So by applying Lemma 18 we get $s_i^L = \frac{3}{2}2^{h_G-i+2} - 2 = 6 \cdot 2^{h_G-i} - 2 = s_i(\phi_A)$ if $i \geq h_G - \left\lfloor \frac{h_G}{2} \right\rfloor + 1$.

The remaining indices are $3 \leq i \leq h_G - \left\lfloor \frac{h_G}{2} \right\rfloor$. Apply Corollary 21 and Lemma 18 to obtain:

$$D \leq \sum_{i=3}^{h_G - \left\lfloor \frac{h_G}{2} \right\rfloor} \left(6 \cdot 2^{h_G-i} - \frac{10}{7}2^{h_G-i+2} \right) = \frac{2}{7}2^{h_G} \sum_{i=3}^{h_G - \left\lfloor \frac{h_G}{2} \right\rfloor} 2^{-i}.$$

The sum of the above geometric progression is $\sum_{i=3}^{h_G - \left\lfloor \frac{h_G}{2} \right\rfloor} 2^{-i} = \frac{1}{4} - \frac{1}{2}2^{h_G - \left\lceil \frac{h_G}{2} \right\rceil} \leq \frac{1}{4} - \frac{\sqrt{2}}{2}2^{-\frac{h_G}{2}}$. Summarizing we get

$$D \leq \frac{2}{7}2^{h_G} \left(\frac{1}{4} - \frac{\sqrt{2}}{2}2^{-\frac{h_G}{2}} \right) = \frac{1}{14}2^{h_G} - \frac{\sqrt{2}}{7}2^{-\frac{h_G}{2}}.$$

By applying Lemma 10 we get the following approximation ratio ρ

$$\rho(h_G) := \frac{OV(G, 2, \phi_A)}{OV(G, 2, \phi_A) - 2D} =$$

$$\frac{\frac{29}{3} \cdot 2^{h_G} - 4h_G - 9 + \frac{1}{3}(-1)^{h_G}}{\left(\frac{29}{3} \cdot 2^{h_G} - 4h_G - 9 + \frac{1}{3}(-1)^{h_G}\right) - 2 \left(\frac{1}{14}2^{h_G} - \frac{\sqrt{2}}{7}2^{-\frac{h_G}{2}}\right)}.$$

After some standard algebraic transformations we obtain

$$\rho(h_G) := \frac{\frac{29}{3}2^{h_G} - 4h_G - \frac{26}{3}}{\frac{200}{21}2^{h_G} - 4h_G + \frac{2\sqrt{2}}{7}2^{\frac{h_G}{2}} - \frac{28}{3}}.$$

Finally, it is not difficult to verify that $\rho(h_G)$ is a strictly monotone increasing sequence (for $h_G \geq 4$) and since

$$\rho := \lim_{h_G \rightarrow +\infty} \rho(h_G) = \lim_{h_G \rightarrow \infty} \frac{\frac{29}{3}2^{h_G} - 4h_G - \frac{26}{3}}{\frac{200}{21}2^{h_G} - 4h_G + \frac{2\sqrt{2}}{7}2^{\frac{h_G}{2}} - \frac{28}{3}} = \frac{203}{200} = 1.015,$$

this completes the proof. \square

6 The complexity of the DAPT with a tree as a guest graph

In this section we show that the DAPT where the guest graph G is a tree on n vertices and the host graph T is a complete d -regular tree of height $\lceil \log_d n \rceil$, for some fixed $d \in \mathbb{N}$, $d \geq 2$, is \mathcal{NP} -hard. This result also settles a more general open question posed by LUCZAK and NOBLE [8] about the complexity of the GEP when both input graphs are trees.

First let us state a simple result on the optimal value of the objective function of the $DAPT(G, d)$ in the case where G is a star graph; this result was proven in in ÇELA and STANĚK [2].

Lemma 24. *Let $G = (V, E)$ be a star graph (i.e. a complete bipartite graph with 1 vertex in one side of the partition and the rest of the vertices in the other side) with n vertices and the central vertex v_1 . Let the host graph T be a complete d -regular tree of height $h = \lceil \log_d n \rceil$, with $d \in \mathbb{N}$, $2 \leq d \leq n$. Then the optimal value of the objective function $OPT(G, d)$ is given by*

$$OPT(G, d) = 2 \left(h n - \frac{d^h - 1}{d - 1} \right). \quad (32)$$

Moreover, an arrangement is optimal if and only if it arranges the central vertex v_1 together with other $d^{h-1} - 1$ arbitrarily selected vertices of G at the leaves of some (arbitrarily selected) basic subtree of T (and the other vertices arbitrarily).

Proof. See ÇELA and STANĚK [2]. \square

Next we will consider another very special case where the guest graph G is the disjoint union of three star graphs.

Lemma 25. *Let the guest graph G be the disjoint union of three star graphs S_i , $1 \leq i \leq 3$, i.e. $V(G) = \dot{\bigcup}_{i=1}^3 V(S_i)$ and $E(G) = \dot{\bigcup}_{i=1}^3 E(S_i)$. Assume that $|V(S_i)| =: n_i$, $1 \leq i \leq 3$, and $n_1 \geq n_2 \geq n_3$, where $V(S_i)$ is the vertex set of S_i , $1 \leq i \leq 3$. Assume that $n := |V(G)| = n_1 + n_2 + n_3$ is a power of d for some fixed $d \in \mathbb{N}$, $d \geq 2$, and $n_1 \geq \frac{n}{d}$. Let the host graph T be a complete d -regular tree of height $h = \log_d n$. Then the optimal value of the objective function $OPT(G, d)$ is given by*

$$OPT(G, d) = 2 \left(h_1 n_1 - \frac{d^{h_1} - 1}{d - 1} \right) + 2 \left(h_2 n_2 - \frac{d^{h_2} - 1}{d - 1} \right) + 2 \left(h_3 n_3 - \frac{d^{h_3} - 1}{d - 1} \right), \quad (33)$$

where $h_i = \lceil \log_d n_i \rceil$, $i \in \{1, 2, 3\}$.

Proof. Let $B = \{b_1, b_2, \dots, b_n\}$ be the set of the leaves of T labelled according to the canonical order. Arrange the central vertex of S_1 together with other $2^{h-1} - 1$ vertices of S_1 at the leaves $b_1, b_2, \dots, b_{d^{h-1}}$ of the leftmost basic subtree of T . The other vertices of S_1 will be arranged later at other appropriately chosen leaves of T . Notice, however, that independently on the arrangement of these vertices the contribution of the edges of S_1 to the objective function value of $DAPT(G, d)$ will be equal to $2 \left(h n_1 - \frac{d^h - 1}{d - 1} \right)$, according to Lemma 24.

Arrange the vertices of S_2 to the n_2 leaves b_{n-n_2+1}, \dots, b_n of T with the largest indices. According to Lemma 24 the edges of S_2 will then contribute by $2 \left(h_2 n_2 - \frac{d^{h_2} - 1}{d - 1} \right)$ to the objective function value of $DAPT(G, d)$. Next, arrange the vertices of S_3 to the n_3 leaves $b_{d^{h-1}+1}, \dots, b_{n_1+n_2+n_3}$ of T (which are still free because $n_1 + n_2 + n_3 = d^h$ and only the leaves with the $d^{h-1} \leq n_1$ smallest indices as well as the leaves with the n_3 largest indices have been occupied already). According to Lemma 24 the edges of S_2 will then contribute by $2 \left(h_3 n_3 - \frac{d^{h_3} - 1}{d - 1} \right)$ to the objective function value of $DAPT(G, d)$. Finally arrange the $n_1 - d^{h-1}$ vertices of S_1 not arranged yet to the remaining $d^h - n_2 - n_3 - d^{h-1} = n_1 - d^{h-1}$ leaves. Summarizing, this arrangement yields an objective function value equal to the expression in (33), and is therefore optimal because the following inequality

$$\begin{aligned} OV(G, d, \phi) &= \sum_{(u,v) \in E(G)} d_T(\phi(u), \phi(v)) = \sum_{i=1}^3 \sum_{(u,v) \in E(S_i)} d_T(\phi(u), \phi(v)) \geq \\ &2 \left(h_1 n_1 - \frac{d^{h_1} - 1}{d - 1} \right) + 2 \left(h_2 n_2 - \frac{d^{h_2} - 1}{d - 1} \right) + 2 \left(h_3 n_3 - \frac{d^{h_3} - 1}{d - 1} \right) \end{aligned}$$

holds for any arrangement ϕ of $DAPT(G, d)$ due to Lemma 24. \square

Next we state the main result of this section.

Theorem 26. *The $DAPT$ with a host graph T being a complete d -regular tree is \mathcal{NP} -hard for every fixed $d \geq 2$ even if the guest graph G is a tree.*

Proof. The problem obviously belongs to \mathcal{NP} . The \mathcal{NP} -hardness is proven by means of a reduction from the *numerical matching with target sums* (NMTS) problem. The NMTS is \mathcal{NP} -hard and is defined as follows (see GAREY and JOHNSON [5]): Let three sets $X = \{x_1, x_2, \dots, x_n\}$, $Y = \{y_1, y_2, \dots, y_n\}$ and $Z = \{z_1, z_2, \dots, z_n\}$ of positive integers, with $\sum_{i=1}^n z_i = \sum_{i=1}^n x_i + \sum_{i=1}^n y_i$ with $n \geq 2$ be given. The goal is to decide whether there exist two permutations (j_1, j_2, \dots, j_n) and (k_1, k_2, \dots, k_n) of the indices $\{1, 2, \dots, n\}$, such that $z_i = x_{j_i} + y_{k_i}$ for all $i = 1, 2, \dots, n$ hold.

Consider an instance of the NMTS and a given integer $d \geq 2$. We construct an instance $DAPT(G, d)$ of the DAPT as follows. Let $l_y \in \mathbb{N}$ be the smallest natural number such that $y_i \leq d^{l_y-4}$, for all $1 \leq i \leq n$. Let $l_x \in \mathbb{N}$ be the smallest natural number such that $x_i + y_j + (d-1)d^{l_y-4} < d^{l_x-2}$, for all $1 \leq i, j \leq n$. Finally, let l_z be the smallest natural number such that $z_i \leq d^{l_z} - (d-1)d^{l_z-4} - (d-1)d^{l_x-2}$, for all $1 \leq i \leq n$. Let then $l := \max\{l_x, l_y, l_z\}$, and let $L \in \mathbb{N}$ be the smallest natural number such that $nd^l \leq d^{L-1}$. Define three vertex disjoint star graphs S_i^x , S_i^y and S_i^z , for every $1 \leq i \leq n$, with $|V(S_i^x)| = (d-1)d^{l-2} + x_i$, $|V(S_i^y)| = (d-1)d^{l-4} + y_i$ and $|V(S_i^z)| = d^l - (d-1)d^{l-4} - (d-1)d^{l-2} - z_i$, for $i = 1, 2, \dots, n$. Notice that $\sum_{i=1}^n (|V(S_i^x)| + |V(S_i^y)| + |V(S_i^z)|) = nd^l$.

Let $v_1(S_i^x)$, $v_1(S_i^y)$ and $v_1(S_i^z)$ be the central vertices of the stars graphs introduced above for $1 \leq i \leq n$, respectively. Next introduce the vertices $u_1, u_2, \dots, u_{\hat{n}}$, where $\hat{n} = d^{L-1} - 1$ and one vertex v_1 . Finally consider a family of stars S'_i , $1 \leq i \leq n'$, with d^l vertices each and $n' := (d-1)d^{L-1-l} - n \geq 0$. Denote by $v_1(S'_i)$, $1 \leq i \leq n'$, their central vertices respectively.

The guest graph $G = (V, E)$ is defined in the following way. The vertex set is given by all vertices defined above, and the edge set contains all edges contained in the stars S_i^x , S_i^y and S_i^z , $1 \leq i \leq n$, and S'_i , $1 \leq i \leq n'$, together with edges connecting the vertex v_1 with the central vertices $v_1(S_i^x)$, $v_1(S_i^y)$, $v_1(S_i^z)$, $1 \leq i \leq n$, and $v_1(S'_i)$, $1 \leq i \leq n'$, and with all vertices $u_1, u_2, \dots, u_{\hat{n}}$. Thus we have

$$V = \left[\bigcup_{i=1}^n [V(S_i^x) \cup V(S_i^y) \cup V(S_i^z)] \right] \cup \left[\bigcup_{i=1}^{n'} V(S'_i) \right] \cup \{u_1, u_2, \dots, u_{\hat{n}}, v_1\} \text{ and}$$

$$E = E_1 \cup E_2 \cup E_3, \text{ where } E_1 = \left\{ (v_1, u_i) : 1 \leq i \leq \hat{n} \right\},$$

$$E_2 = \left[\bigcup_{i=1}^{n'} E(S'_i) \right] \cup \left\{ (v_1, v_1(S'_i)) : 1 \leq i \leq n' \right\} \text{ and}$$

$$E_3 = \bigcup_{i=1}^n \left[E(S_i^x) \cup E(S_i^y) \cup E(S_i^z) \cup \left\{ (v_1, v_1(S_i^x)), (v_1, v_1(S_i^y)), (v_1, v_1(S_i^z)) \right\} \right].$$

The number of vertices in G is given as $|V(G)| = \sum_{i=1}^n (|V(S_i^x)| + |V(S_i^y)| + |V(S_i^z)|) + \widehat{n} + n'd^l + 1 = d^L$. Thus the host graph is a d -regular tree T of height $h = \lceil \log_d |V(G)| \rceil = L$.

We show that the optimal value $OPT(G, d)$ of the objective function of $DAPT(G, d)$ equals the expression in (35) if and only if the corresponding NMTS instance is a YES-instance, and this would complete the \mathcal{NP} -hardness proof.

Prior to showing the above if and only if statement, consider the optimal arrangement of the substar G' induced in G by v_1 and its neighbours, i.e. by the following set of vertices

$$\{v_1\} \cup \{u_i : 1 \leq i \leq \widehat{n}\} \cup \{v_1(S'_i) : 1 \leq i \leq n'\} \cup \left[\cup_{i=1}^n \{v_1(S_i^x), v_1(S_i^y), v_1(S_i^z)\} \right].$$

Assume w.l.o.g. that the vertex v_1 is arranged at the most left leaf of T (i.e. the first leaf b_1 of T in the canonical ordering). The vertex v_1 has $3n + \widehat{n} + n' = d^{L-1} - 1 + 3n + n' > d^{L-1} - 1$ neighbours (note that $n' \geq 0$) and hence $|V(G')| = 3n + \widehat{n} + n' + 1 = d^{L-1} + 3n + n'$. According to Lemma 24, any arrangement of G' which arranges the $d^{L-1} - 1$ neighbours $\{u_1, u_2, \dots, u_{\widehat{n}}\}$ of v_1 at the leaves of the leftmost basic subtree of T and the remaining $3n + n'$ neighbours at some other (arbitrarily selected) leaves of T is optimal. In particular such an arrangement would not arrange $v_1(S_i^x)$, $v_1(S_i^y)$, $v_1(S_i^z)$, $i = 1, 2, \dots, n$, and $v_1(S'_i)$, $1 \leq i \leq n'$, at the leaves of the leftmost basic subtree.

Let us now proof the if and only if statement formulated above.

The “if” statement.

Assume that the NMTS instance is a YES-instance. We show that the equality (35) holds. Consider two permutations (j_1, j_2, \dots, j_n) and (k_1, k_2, \dots, k_n) of the indices $\{1, 2, \dots, n\}$, such that $z_i = x_{j_i} + y_{k_i}$, for all $i = 1, 2, \dots, n$. Then for all $i \in \{1, 2, \dots, n\}$ we have

$$\begin{aligned} |V(S_{j_i}^x)| + |V(S_{k_i}^y)| + |V(S_i^z)| &= (d-1)d^{l-2} + x_{j_i} + (d-1)d^{l-4} + y_{k_i} + d^l - \\ (d-1)d^{l-4} - (d-1)d^{l-2} - z_i &= d^l. \end{aligned} \quad (34)$$

Thus, for every $i \in \{1, 2, \dots, n\}$ the disjoint stars graphs $S_{j_i}^x$, $S_{k_i}^y$ and S_i^z can be optimally arranged at the leaves of the i -th rightmost subtree of l -th order according to Lemma 25. Notice that the assumptions of the lemma are fulfilled because $|V(S_i^z)| > (d-1)d^{l-1} \geq d^{l-1}$. (Indeed, due to $y_{k_i} \leq d^{l-4}$ and $x_{j_i} + y_{k_i} + (d-1)d^{l-4} < d^{l-2}$ we get $|V(S_{k_i}^y)| + |V(S_{j_i}^x)| = (d-1)d^{l-4} + y_{k_i} + (d-1)d^{l-2} + x_{j_i} < d^{l-1}$ which together with (34) implies $|V(S_i^z)| > (d-1)d^{l-1} \geq d^{l-1}$.) Since $nd^l \leq d^{L-1}$ the n rightmost subtrees of the l -th order are subtrees of the rightmost basic subtree (of height $L-1$).

It remains to arrange the vertices $v_1, u_1, u_2, \dots, u_{\widehat{n}}$ and the stars S'_i , $1 \leq i \leq n'$. This is done by arranging $v_1, u_1, u_2, \dots, u_{\widehat{n}}$ at the leaves of the leftmost basic subtree and the stars S'_i , $1 \leq i \leq n'$ in one of the n' still free subtrees of l -th order each.

These arrangements are clearly optimal for each of the stars S'_i , $1 \leq i \leq n'$ (recall that they have d^l vertices each). Moreover, as mentioned above this is also an optimal arrangement of G' . Thus this arrangement arranges optimally the following subgraphs of G : G' , S'_i , $1 \leq i \leq n'$, and the disjoint unions of the triples $(S_{j_i}^x, S_{k_i}^y, S_i^z)$, for $1 \leq i \leq n$, respectively. Since the edge sets of the above mentioned graphs yield a partition of the edge set $E(G)$, the arrangement described above is an optimal arrangement of G . The corresponding value $OPT(G, d)$ of the objective function is given as the sum of $OPT(G', d)$, $OPT(S'_i, d)$, $1 \leq i \leq n'$, and $OPT(S_{j_i}^x \cup S_{k_i}^y \cup S_i^z, d)$, for $1 \leq i \leq n$. According to Lemma 24 and Lemma 25 we get

$$OPT(G, d) = 2\left(Ln'' - \frac{d^L - 1}{d - 1}\right) + 2n'\left(d^l - \frac{d^l - 1}{d - 1}\right) +$$

$$2 \sum_{i=1}^n \left[l|V(S_i^z)| - \frac{d^{l-1}}{d-1} + (l-1)|V(S_{j_i}^x)| - \frac{d^{l-1} - 1}{d-1} + (l-3)|V(S_{k_i}^y)| - \frac{d^{l-3} - 1}{d-1} \right], \quad (35)$$

where $n'' := |V(G')| = \hat{n} + n' + d^{L-1}$.

The “only if” statement. Assume that $OPT(G, d)$ is given as in (35) and consider some optimal arrangement ϕ of G . We show that the corresponding NMTS instance is a YES-instance. Notice that (35) implies

$$OPT(G, d) = OPT(G', d) + \sum_{i=1}^{n'} OPT(S'_i, d) +$$

$$\sum_{i=1}^n (OPT(S_i^x, d) + OPT(S_i^y, d) + OPT(S_i^z, d)).$$

Thus, in particular, ϕ yields optimal arrangements of G' , S'_i , $1 \leq i \leq n'$, and S_i^x , S_i^y and S_i^z , $1 \leq i \leq n$. Assume w.l.o.g. that ϕ arranges v_1 at the leftmost leaf of T . Then according to Lemma 24 ϕ arranges neighbours of v_1 , i.e. vertices of G' , to each leaf of the leftmost basic subtree of T . Thus, none of the neighbours of v_1 arranged at the leaves of the leftmost basic subtree of T can be the central vertex of some of the stars S'_i , $1 \leq i \leq n'$, or S_i^x , S_i^y and S_i^z , $1 \leq i \leq n$, because then according to Lemma 24 ϕ would not lead to an optimal arrangement of the corresponding star. It follows that the vertices arranged at the leaves of the leftmost basic subtree of T are u_i , $1 \leq i \leq \hat{n}$.

According to Lemma 24 ϕ arranges each star S'_i , $1 \leq i \leq n'$, to the leaves of some subtree of l -th order. Hence there remain $(d-1)d^{L-1-l} - n' = n$ subtrees of l -th order at the leaves of which ϕ the stars S_i^x , S_i^y and S_i^z , $1 \leq i \leq n$.

Recall now that $(d-1)d^{l-1} < |V(S_i^z)|$, $(d-1)d^{l-2} < |V(S_i^x)| < d^{l-1}$ and $(d-1)d^{l-4} < |V(S_i^y)| < d^{l-3}$ for $1 \leq i \leq n$. Thus in each of the n remaining free

subtrees of l -th order can not be arranged more than one of the stars S_i^z , $1 \leq i \leq n$, and since there are n such stars to be arranged in n subtrees of l -th order exactly one of them will be arranged in each subtree. By analogous arguments we get that exactly one of the stars S_i^x will be arranged in each of the subtrees of l -th order mentioned above, and finally, exactly one of the stars S_i^y will be arranged in each of these subtrees. Thus in each of the n -th subtrees of l -th level exactly one star S_i^z , one star S_i^x and one star S_i^y will be arranged. For all $i \in \{1, 2, \dots, n\}$ denote by j_i and k_i the indices of the stars arranged together with S_i^z in the same subtree, i.e. $S_{j_i}^x$, $S_{k_i}^y$ and S_i^z are arranged in the same subtree of l -th order, $1 \leq i \leq n$. Clearly, (j_1, j_2, \dots, j_n) and (k_1, k_2, \dots, k_n) are permutations of $\{1, 2, \dots, n\}$, respectively. Since the stars S_i^x , S_i^y and S_i^z , $1 \leq i \leq n$, have nd^l vertices altogether, which is the number of leaves of the n subtrees of l -th level, the equality

$$|V(S_{j_i}^x)| + |V(S_{k_i}^y)| + |V(S_i^z)| = d^l \quad (36)$$

must hold, for all $1 \leq i \leq n$. By substituting the cardinalities of the vertex sets of the stars in (36) we get $x_{j_i} + y_{k_i} - z_i = 0$, for all $1 \leq i \leq n$, and this completes the proof. □

7 Final notes, conclusions and outlook

This paper deals with a special case of the data arrangement problem on regular trees (DAPT), namely with the case where both the guest and the host graph are binary regular trees of heights h_G and $h = h_G + 1$, respectively. Some basic properties of the problem are identified and an approximation algorithm with approximation ratio 1.015 is proposed. Moreover, we provide closed formulas for the arrangement generated by the approximation algorithm mentioned above and for its corresponding objective function value.

The analysis of the approximation algorithm and the estimation of the approximation ratio involve a special case of the *k-balanced partitioning problem (k-BPP)* as an auxiliary problem. More precisely we consider the *k-BPP* where the input graph is a binary regular tree for a particular choice of k , namely for k being a power of 2, and give (a lower bound for the) optimal value its objective function.

Further we investigate the relationship between the considered particular cases of the DAPT and the *k-BPP* and derive a lower bound for the value of the objective function of the $DAPT(G, 2)$ by solving h_G instances of the $2^{k'}\text{-BPP}$, where h_G is the height of the guest graph G and $k' = 1, 2, \dots, h_G$. This lower bound leads then to the above mentioned approximation ratio.

It would be interesting to investigate whether some alternative analysis of the proposed algorithm for the $DAPT(G, 2)$ could lead to a better approximation ratio. Numerical results provide some evidence that the answer to this question might be “yes” and suggest an empirical approximation ratio of 1.0098.

Finally we settle an open question from the literature and prove that the DAPT with a host graph being a d -regular tree, $d \geq 2$, $d \in \mathbb{N}$, is \mathcal{NP} -hard even if the guest graph is a tree.

The complexity of the DAPT in the case where both the guest and the host graph are binary regular trees remains an open question for further research. Other open questions concerns the more general cases of the DAPT where the guest graph G is a d -regular tree and the host graph T is a d' -regular tree with $d = d' \geq 3$ or $d \neq d'$.

Acknowledgements

The research was funded by the Austrian Science Fund (FWF): P23829.

References

- [1] K. Andreev and H. Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.
- [2] E. Çela and R. Staněk. Heuristics for the data arrangement problem on regular trees. *Journal of Combinatorial Optimization*, 30(3):768–802, 2015.
- [3] F. R. K. Chung. On optimal linear arrangements of trees. *Computers & Mathematics with Applications*, 10(1):43–60, 1984.
- [4] A. E. Feldmann and L. Foschini. Balanced partitions of trees and applications. *Algorithmica*, 71(2):354–376, 2015.
- [5] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Series of books in the mathematical sciences, 1979.
- [6] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153–168, 1992.
- [7] R. Krauthgamer, J. S. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *Proceeding SODA '09 Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 942–949, 2009.
- [8] M. Luczak and S. Noble. Optimal arrangement of data in a tree directory. *Discrete Applied Mathematics*, 121(1–3):307–315, 2002.
- [9] Y. Shiloach. A minimum linear arrangement algorithm for undirected trees. *SIAM Journal on Computing*, 8(1):15–22, 1979.

A Appendix

A.1 Arrangements ϕ_A obtained from Algorithm 1 for different heights h_G of the guest graph G



Figure 13: Guest graph $G = (V, E)$ (binary regular tree of height $h_G = 0$).

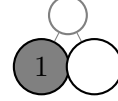


Figure 14: Arrangement ϕ_A obtained from Algorithm 1 for the guest graph of height $h_G = 1$ depicted in Figure 13. Its objective function value is $OV(G, 2, \phi_A) = 0$.

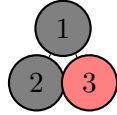


Figure 15: Guest graph $G = (V, E)$ (binary regular tree of height $h_G = 1$).

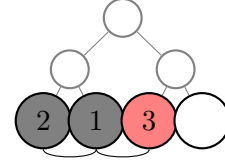


Figure 16: Arrangement ϕ_A obtained from Algorithm 1 for the guest graph of height $h_G = 1$ depicted in Figure 15. Its objective function value is $OV(G, 2, \phi_A) = 6$.

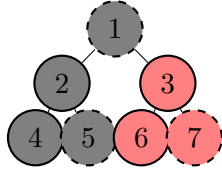


Figure 17: Guest graph $G = (V, E)$ (binary regular tree of height $h_G = 2$).

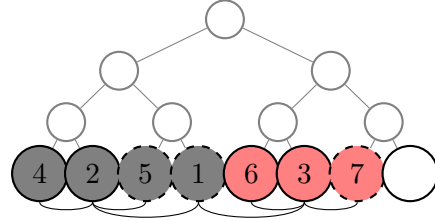


Figure 18: Arrangement ϕ_A obtained from Algorithm 1 for the guest graph of height $h_G = 2$ depicted in Figure 17. Its objective function value is $OV(G, 2, \phi_A) = 22$.